

**Dana Lica**

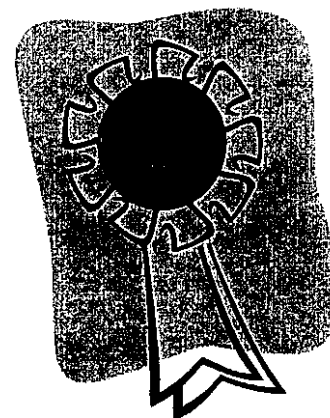
**Mircea Pașoi**

# **INFORMATICĂ**

## **FUNDAMENTELE PROGRAMĂRII**

**Ediție revizuită și adăugită**

Culegere de probleme – Pascal și C/C++  
pentru clasa a IX-a



*Editura L&S SOFT*

Toate drepturile asupra acestei lucrări aparțin editurii L&S SOFT.

Reproducerea integrală sau parțială a textului din această carte este posibilă doar cu acordul în scris al editurii L&S SOFT.

**Descrierea CIP a Bibliotecii Naționale a României**  
**LICA, DANA**

**Informatică : fundamentele programării : culegere de probleme pentru clasa a IX-a / Dana Lica, Mircea Pașoi. - București : Editura L&S Soft, 2006**  
ISBN 973-86022-9-7

I. Pașoi, Mircea

**Editura L&S SOFT:**

**Telefon:** 0722-573701; 0727-731.947;

**E-mail:** office@ls-infomat.ro

**Web Site:** www.ls-infomat.ro

Tiparul executat la **S.C. Lumina TIPO s.r.l.**

Str. Luigi Galvani nr. 20 bis, Sector. 4, București, tel./fax: 211.32.60; tel: 212.29.27

E-mail: office@luminatipo.com; www.luminatipo.com

## *Cuprins*

### Capitolul 1

#### Programarea structurată și instrucțiuni în limbajul de programare Pascal / C / C ++

1.1 Structura liniară și alternativă – Instrucțiunea de atribuire și condițională.....	5
1.1.1 Teste cu alegere multiplă și duală.....	5
1.1.2 Teste cu itemi semiobiectivi.....	18
1.1.3 Probleme rezolvate.....	22
1.1.4 Probleme propuse.....	27
1.2 Structuri repetitive – instrucțiuni repetitive.....	31
1.2.1 Teste cu alegere multiplă și duală.....	31
1.2.2 Teste cu itemi semiobiectivi.....	44
1.2.3 Probleme rezolvate.....	49
1.2.4 Probleme propuse.....	57
1.3 Probleme de concurs ce procesează date simple.....	64
1.3.1 Probleme rezolvate.....	64
1.3.2 Probleme propuse.....	70

### Capitolul 2

#### Tipuri de date structurate

2.1 Tabloul unidimensional.....	81
2.1.1 Teste cu alegere multiplă și duală.....	81
2.1.2 Teste cu itemi semiobiectivi.....	84
2.1.3 Probleme rezolvate.....	89
2.1.4 Probleme propuse.....	99
2.2 Tabloul bidimensional.....	106
2.2.1 Teste cu alegere multiplă și duală.....	106
2.2.2 Teste cu itemi semiobiectivi.....	111
2.2.3 Probleme rezolvate.....	115
2.2.4 Probleme propuse.....	122

2.3 Fișiere text	129
2.3.1 Teste cu alegere multiplă și duală	129
2.3.2 Probleme rezolvate	132
2.3.3 Probleme propuse	139
2.4 Probleme de concurs ce procesează date structurate	141
2.4.1 Probleme rezolvate	141
2.4.2 Probleme propuse	152
<b>Indicații și răspunsuri</b>	172

## CAPITOLUL 1

### Programarea Structurată și Instrucțiuni în Limbajul de Programare Pascal / C / C++

#### 1.1. Structura liniară și alternativă – Instrucțiunea de atribuire și condițională

##### 1.1.1 Teste cu alegere multiplă și duală

1. Care dintre următoarele valori fac parte din tipul întreg:

- |          |          |
|----------|----------|
| a) 23.0  | d) +1234 |
| b) -4321 | e) 12345 |
| c) -24.0 | f) 0.0   |

2. Care dintre următoarele operații au ca rezultat valori din tipul întreg sau real ? (Operatorul [] desemnează partea întreagă, iar operatorul mod desemnează restul la împărțirea întreagă):

- |                        |                      |
|------------------------|----------------------|
| a) $3 \neq 4$          | d) $2 \bmod 10$      |
| b) $[10.345]$          | e) $10.01 < 14.5$    |
| c) $\text{not}(5 > 6)$ | f) $\sqrt{10.0 + 6}$ |

3. Care dintre următoarele operații sunt corecte sintactic:

- |                       |                             |
|-----------------------|-----------------------------|
| a) $10 + 2.3$         | d) $\text{true and } 1.0;$  |
| b) $\text{not true};$ | e) $\text{not true false};$ |
| c) $23 \bmod 2.0$     | f) $23.45 < 17;$            |

4. Care dintre următoarele operații au ca rezultat valoarea true știind că variabilele întregi  $a$  și  $b$  au valorile  $a = 23$  și  $b = 50$ :

- |                |                  |
|----------------|------------------|
| a) $a \neq b;$ | d) $b \leq a;$   |
| b) $a > b;$    | e) $a \bmod 10;$ |
| c) $a + 10;$   | f) $b \geq a;$   |

5. Care dintre următoarele operații au ca rezultat valoarea 3 știind că variabilele întregi  $a$  și  $b$  au valorile  $a = 45$  și  $b = 120$ :

- |                  |                         |
|------------------|-------------------------|
| a) $a \bmod 6;$  | d) $b \text{ div } 39;$ |
| b) $a \bmod 10;$ | e) $b - 2 * a;$         |
| c) $a - 15;$     | f) $a \bmod 7;$         |

6. Care dintre următoarele operații au ca rezultat valoarea 10.0 știind că variabilele reale  $x, y$  și  $z$  au valorile  $x = 20.0, y = 15.0$  și  $z = 1.5$ :

- |               |                 |
|---------------|-----------------|
| a) $y/z$      | d) $x-y/z$      |
| b) $y*z$      | e) $\sqrt{x*z}$ |
| c) $\sqrt{x}$ | f) $x - x/2$    |

7. Care dintre următoarele expresii sunt corecte sintactic?

- |                           |                               |
|---------------------------|-------------------------------|
| a) $18 - 3 + 2$           | e) $3 + 4 \text{ mod } 2.3$   |
| b) $18 + 7/3 + 2$         | f) $24 \text{ div } (7 + 1)$  |
| c) $2.30 + 3 \text{ mod}$ | g) $24 \text{ div not}(7/3)$  |
| d) $3 + 7.0/3$            | h) $3 \text{ mod div } 4 - 1$ |

8. Care dintre următoarele expresii sunt întregi, dacă toate variabilele care intervin sunt de tip întreg?

- |                               |  |
|-------------------------------|--|
| a) $(a+3.5)*2$                | d) $57 \text{ mod } a < a \text{ mod } 57$ |
| b) $a*3 \text{ div } 4 - x*5$ | e) $[3.50*x] - 15 \text{ div } 3$          |
| c) $\text{not}(5<98)$         | f) $\sqrt{10} * 2 + 4 \text{ div } a$      |

9. Care dintre următoarele expresii sunt reale, dacă toate variabilele care intervin sunt de tip întreg?

- |                               |                           |
|-------------------------------|---------------------------|
| a) $\text{not}(x*z < z)$      | d) $\sqrt{x*z} * 2 - 32$  |
| b) $x*y \text{ mod } z + x*5$ | e) $[3.50*x] - 123.45*y$  |
| c) $(a*2 + 12.5)*5$           | f) $(x+2) \text{ mod } 3$ |

10. Care dintre următoarele expresii sunt logice știind că toate variabilele care intervin sunt de tip numeric (întreg sau real)? Considerăm expresiile ca fiind corecte.

- |  |  |
|--|--|
| a) $(a \neq x) \text{ or } (34 > (x - y))$ | d) $x*y \text{ mod } a < a \text{ mod } x$ |
| b) $a/x/y - x*y$                           | e) $\text{not}([3.50] < 5)$                |
| c) $(a < (b+x)) \text{ and } (3 < 56)$     | f) $\sqrt{x+y} * 3 + 7 \text{ div } a$     |

11. Care este valoarea expresiei:  $20 \text{ div } 10 * 2 + 30 \text{ div } 15 * 2$

- |      |      |      |      |
|------|------|------|------|
| a) 2 | b) 0 | c) 8 | d) 4 |
|------|------|------|------|

12. Care este valoarea expresiei:  $4000/10/10*2 + 4*10*10/2$

- |          |          |          |          |
|----------|----------|----------|----------|
| a) 400.0 | b) 280.0 | c) 240.0 | d) 220.0 |
|----------|----------|----------|----------|

13. Asociați operatorilor din coloana dreaptă operațiile corespunzătoare din coloana stângă:

- |        |        |                   |                      |
|--------|--------|-------------------|----------------------|
| a) /   | e) mod | 1) Înmulțire      | 5) Cât la împărțire  |
| b) div | f) []  | 2) Scădere        | 6) Conjunție         |
| c) *   | g) and | 3) Parte întreagă | 7) Rest la împărțire |
| d) -   | h) not | 4) Negație        | 8) Împărțire         |

14. Care dintre următoarele secvențe sunt echivalente (conduc la obținerea aceluiași rezultat) cu instrucțiunea:

$$a \leftarrow (a+b+c)/2;$$

- |                                    |  |
|------------------------------------|--|
| a) $a \leftarrow b/2 + a/2 + c/2;$ | d) $a \leftarrow b/1/2 + a/1/2 + c/1/2;$ |
| b) $a \leftarrow a + b + c/2;$     | e) $a \leftarrow a + b/2 + c/2;$         |
| c) $a \leftarrow (a + b)/2 + c/2;$ | f) $a \leftarrow (a + b)/1/2 + c/1/2;$   |

15. Care sunt valorile variabilelor întregi  $a$  și  $b$  după execuția instrucțiunilor, dacă inițial ele aveau valorile  $a=82$  și  $b=24$ :

- |  |                       |
|--|-----------------------|
| $a \leftarrow a \text{ mod } 2 + b \text{ div } 2 \text{ div } 2;$ | a) $a = 5$ și $b = 5$ |
| $b \leftarrow 2 * a \text{ mod } 2;$                               | b) $a = 6$ și $b = 6$ |
| $a \leftarrow a + b;$  | c) $a = 0$ și $b = 6$ |
| $b \leftarrow a \text{ mod } 2 + b + 10 \text{ mod } 2;$           | d) $a = 6$ și $b = 0$ |
|  | e) $a = 0$ și $b = 0$ |

16. Care dintre variabilele  $a, b, c, d$  vor avea aceeași valoare după execuția instrucțiunilor următoare, dacă inițial ele aveau valorile  $a=10, b=20, c=30$  și  $d=40$ :

- |   |                                 |
|---|---------------------------------|
| $a \leftarrow [\sqrt{d*2+b}];$                  | a) $a = 100, d = 100$           |
| $b \leftarrow a + b \text{ div } 2 * 5;$        | b) $a = 60, c = 60$             |
| $c \leftarrow (b - d) * 5 \text{ div } 10;$     | c) $a = 10, c = 10$ și $d = 10$ |
| $d \leftarrow (d + a + b - c) \text{ div } 10;$ | d) $b = 60, d = 60$             |

17. Determinați ordinea de executare a instrucțiunilor următoare pentru ca la final variabilele  $x, y$ , și  $z$  să aibă valori egale, indiferent de valorile avute anterior:

- |                                      |                  |
|--------------------------------------|------------------|
| a) $x \leftarrow x \text{ mod } 15$  | a) $b, d, a, c;$ |
| b) $z \leftarrow x \text{ div } y;$  | b) $c, d, b, a;$ |
| c) $x \leftarrow 100;$               | c) $a, b, d, c;$ |
| d) $y \leftarrow x \text{ div } 10;$ | d) $c, a, b, d;$ |

18. Care dintre următoarele atribuiri fac ca valoarea variabilei reale  $x$  să aibă partea fracționară egală cu 0.0, indiferent de valoarea inițială a acesteia? (Operația parte întreagă este desemnată prin operatorul [])

- |                             |                                |
|-----------------------------|--------------------------------|
| a) $x \leftarrow x*10;$     | d) $x \leftarrow x/10;$        |
| b) $x \leftarrow [x] * 10;$ | e) $x \leftarrow [x] + 10.03;$ |
| c) $x \leftarrow [x*10];$   | f) $x \leftarrow [x + 10];$    |

19. Care dintre următoarele atribuiri sunt corecte știind că variabilele  $a$  și  $b$  sunt întregi, iar variabilele  $x$  și  $y$  sunt reale:

- |   |  |
|---|--|
| a) $x \leftarrow a * 10 \text{ div } 3$ ; | d) $b \leftarrow (x + y) \text{ div } 10 + a \text{ mod } 2$ ; |
| b) $a \leftarrow [x] * b \text{ mod } 2$  | e) $x \leftarrow a + b/3$ ;                                    |
| c) $y \leftarrow x \text{ mod } 2 + 1$ ;  | f) $y \leftarrow [x + a] \text{ div } 3$ ;                     |

20. Care dintre următoarele operații au ca rezultat valoarea True știind că variabilele întregi  $a$  și  $b$  au valorile  $a = 23$  și  $b = 50$ :

- |  |  |
|--|--|
| a) $(a \neq b) \text{ and } (a > b)$ ;       | d) $b \text{ mod } 10 > a \text{ div } 7$ ;          |
| b) $((a+10) < b) \text{ or } \text{false}$ ; | e) $\text{not false and } (a \text{ div } 10 < b)$ ; |
| c) $\text{true and } (a \neq b)$             | f) $\text{not (true or (a+b < 10))}$ ;               |

21. Care sunt valorile variabilelor  $x$ ,  $y$  și  $z$  în urma executării secvenței de instrucțiuni (Operatorul  $\leftrightarrow$  desemnează operația de interschimbare a valorilor a două variabile):

- |   |                             |
|---|-----------------------------|
| $x \leftarrow 1234$ ;                   | a) $x = 12, y = 18, z = 24$ |
| $y \leftarrow x \text{ div } 100$ ;     | b) $x = 12, y = 24, z = 18$ |
| $x \leftarrow x * 2 \text{ div } 100$ ; | c) $x = 34, y = 18, z = 24$ |
| $z \leftarrow (x+y) \text{ div } 2$ ;   | d) $x = 18, y = 12, z = 24$ |
| $z \leftrightarrow x$                   |                             |

22. Care dintre operațiile următoare atribuie variabilei reale  $x$  media aritmetică a valorilor variabilelor întregi  $a$ ,  $b$  și  $c$ :

- |   |                                     |
|---|-------------------------------------|
| a) $x \leftarrow (a + b + c) / 2$ ;       | d) $x \leftarrow a + b/3 + c/3$ ;   |
| b) $x \leftarrow a/3 + b/3 + c/3$ ;       | e) $x \leftarrow a/3 + b/2 + c/2$ ; |
| c) $x \leftarrow a/1/3 + b/1/3 + c/1/3$ ; | f) $x \leftarrow (a + b + c) / 3$ ; |

23. Care dintre operațiile următoare atribuie variabilei întregi  $x$  una din cifrele sale, știind că  $x > 10000$ :

- |   |  |
|---|--|
| a) $x \leftarrow x \text{ mod } 100$ ;                | d) $x \leftarrow x \text{ div } 100 \text{ mod } 10$ ; |
| b) $x \leftarrow x \text{ mod } 10$ ;                 | e) $x \leftarrow x \text{ mod } 10 \text{ div } 1$ ;   |
| c) $x \leftarrow x \text{ div } 10 \text{ mod } 10$ ; | f) $x \leftarrow x \text{ mod } 50$ ;                  |

24. Care dintre operațiile de atribuire următoare sunt corecte, știind că toate variabilele au tipul întreg?

- |  |   |
|--|---|
| a) $a \leftarrow a + 3$                | d) $x \leftarrow a < 2$                               |
| b) $x \leftarrow a + 3 - 2$            | e) $4 + a \leftarrow b$                               |
| c) $b \leftarrow b \text{ div } 2 + 1$ | f) $a \leftarrow x \text{ div } 10 \text{ mod } 10$ ; |

25. Variabila  $max$  are valoarea 3. Care dintre următoarele operații de atribuire permite ca variabila  $max$  să își modifice valoarea din 3 în 0?

- a)  $max \leftarrow max - 12 \text{ div } 4$   
b)  $max \leftarrow (max - 1) + 3$

- c)  $max \leftarrow 2 * max - 6$   
d)  $max \leftarrow 2 + 5 \text{ mod } 2$

26. Care sunt valorile variabilelor  $x$  și  $y$  după executarea în ordine a următoarelor trei operații de atribuire?

- $x \leftarrow 3$ ;  
 $y \leftarrow x + 3$ ;  
 $x \leftarrow x - 3$ ;

- a)  $x = 0$  și  $y = 3$   
b)  $x = 3$  și  $y = 0$   
c)  $x = 6$  și  $y = 0$   
d)  $x = 0$  și  $y = 6$

27. Care sunt valorile variabilelor reale  $x$  și  $y$  după executarea instrucțiunilor următoare:

- $x \leftarrow 22.50$ ;  
 $y \leftarrow [x] + 8$ ;  
 $x \leftarrow 2 * x$ ;  
 $y \leftarrow y + x$ ;

- a)  $x = 35.0$  și  $y = 25.0$   
b)  $x = 45.0$  și  $y = 75.0$   
c)  $x = 28.0$  și  $y = 30.0$   
d)  $x = 22.5$  și  $y = 75.0$

28. Care dintre variabilele care intervin în secvența de operații următoare își vor păstra valoarea avută inițial?

- $a \leftarrow b + c$ ;  
 $c \leftarrow a - c$ ;  
 $b \leftarrow c$ ;  
 $c \leftarrow a - b$ ;

- a)  $a$  și  $c$   
b)  $b$  și  $c$   
c)  $c$  și  $a$   
d)  $a$ ,  $b$ , și  $c$

29. Care sunt valorile variabilelor întregi  $a$  și  $b$  după executarea instrucțiunilor următoare:

- $a \leftarrow 1235$ ;  $b \leftarrow a \text{ mod } 10$ ;  
daca  $(a - b) \text{ mod } 10 = 0$  atunci  
     $a \leftarrow a \text{ div } 100$ ;  
     $b \leftarrow a \text{ mod } 100$ ;  
daca  $a = b$  atunci  $a \leftarrow a * 100$ ;

- a)  $a = 1200$  și  $b = 5$   
b)  $a = 12$  și  $b = 35$   
c)  $a = 3500$  și  $b = 12$   
d)  $a = 1200$  și  $b = 12$   
e)  $a = 1200$  și  $b = 35$   
f)  $a = 100$  și  $b = 35$

30. Care sunt valorile obținute de variabilele întregi  $x$ ,  $y$  și  $z$  după executarea operației de decizie următoare, dacă la intrare aveau valorile  $x = 23$ ,  $y = 14$  și  $z = 25$ ?

- daca  $(x > 1) \text{ and } (y - z > 0)$  atunci  
     $x \leftarrow y - z$   
altfel  
     $y \leftarrow x - 1$ ;  
     $z \leftarrow y + x$ ;

- a)  $x = -11$ ,  $y = 14$ ,  $z = 25$   
b)  $x = 23$ ,  $y = 22$ ,  $z = 37$   
c)  $x = 23$ ,  $y = 22$ ,  $z = 45$   
d)  $x = -11$ ,  $y = 22$ ,  $z = 37$   
e)  $x = 23$ ,  $y = -11$ ,  $z = 25$

31. Care dintre secvențele de operații următoare conduc la afișarea a două valori pozitive?

- |    |   |    |   |
|----|---|----|---|
| a) | daca (a>0) and (b=a) atunci<br>scrie (a, b) | c) | daca (a>0) or (b>0) atunci<br>scrie (a, b)    |
| b) | daca (a*b)>0 atunci<br>scrie (a, b)         | d) | daca (a*b>0) and (b>0) atunci<br>scrie (a, b) |

32. Care dintre următoarele numere reprezintă numere întregi din vocabularul limbajului Pascal /C/C++?

- a) -315.2    b) 1982    c) +23    d) 002222    e) 23E2    f) '123'

33. Care dintre următoarele numere reprezintă numere reale din vocabularul limbajului Pascal /C/C++?

- a) 445.6    b) -45.2    c) 22,17    d) pi    e) 23E2    f) '12.3'

34. Se consideră următorul program:

```
begin
  write('Eu ');
  writeln('sunt ');
  write('bine')
end.

void main() {
  cout << "Eu ";
  cout << "sunt" << endl;
  cout << "bine";
}
```

În ce fel se va face afișarea mesajelor?

- |      |         |              |           |
|------|---------|--------------|-----------|
| a)   | b)      | c)           | d)        |
| Eu   | Eu sunt | Eu sunt bine | Eu        |
| sunt | bine    |              | sunt bine |
| bine |         |              |           |

35. Care dintre următoarele secvențe conduc la o afișare în același format cu cel produs de apelul: `write('ABC')` în Pascal, respectiv `cout << "ABC"` în C/C++?

- |                |                |                   |                |
|----------------|----------------|-------------------|----------------|
| a)             | b)             | c)                | d)             |
| write('A');    | write('AB');   | cout << "A";      | cout << "AB";  |
| write('B');    | writeln('C');  | cout << "B";      | cout << "C" << |
| write('C');    |                | cout << "C";      | endl;          |
| c)             | d)             | c)                | d)             |
| writeln('AB'); | writeln('ABC') | cout<<"AB"<<endl; | cout << "ABC"  |
| write('C');    |                | cout << "C";      | << endl;       |

36. Ce va fi afișat pe ecran în urma apelului `writeln(45.23:6:3)` în Pascal, respectiv `printf("%6.3f\n", 45.23)` în C/C++?

- a) +45.230    b) 45.230    c) 045.230    d) 45.023

37. Care dintre următoarele apeluri sunt incorecte sintactic?

- |              |                |                 |                 |
|--------------|----------------|-----------------|-----------------|
| a)           | b)             | a)              | b)              |
| write("A")   | write('13.45') | cout << 'A'     | cout << "13.45" |
| c)           | d)             | c)              | d)              |
| writeln(1.3) | writeln(3a)    | cout<<1.3<<endl | cout << 3a      |

38. Care dintre următoarele apeluri conduce la afișarea unor caractere ce pot reprezenta un număr întreg?

- |               |               |                  |                 |
|---------------|---------------|------------------|-----------------|
| a)            | b)            | a)               | b)              |
| write(-123)   | write('1234') | cout << -123     | cout << "1234"  |
| c)            | d)            | c)               | d)              |
| writeln(-1.3) | writeln(1.0)  | cout<<-1.3<<endl | cout<<1.0<<endl |

39. Care dintre tipurile următoare reprezintă tipuri de date reale?

- |      |        |              |               |
|------|--------|--------------|---------------|
| a)   | b)     | a)           | b)            |
| real | byte   | float        | unsigned char |
| c)   | d)     | c)           | d)            |
| word | double | unsigned int | double        |

40. Considerăm că variabila `a` are valoarea -13. Cărui tip poate aparține această variabilă?

- |               |                |         |             |
|---------------|----------------|---------|-------------|
| a)            | b)             | a)      | b)          |
| var a:char    | var a:byte     | byte a; | unsigned a; |
| c)            | d)             | c)      | d)          |
| var a:integer | var a:shortint | int a;  | short a;    |

41. Stabiliți care dintre următoarele declarații de variabile sunt corecte:

- |                    |               |
|--------------------|---------------|
| a)                 | b)            |
| var a:integer[10]; | float lt,2t;  |
| var x,y:int;       | int a[1..10]; |
| var a,b:real;      | int x,y;      |
| var lt,2t:integer; | string s;     |
| var z:longinteger; | double e,v;   |

42. Dacă `a, b` sunt variabile de tip `integer` (varianta Pascal) / `int` (varianta C/C++), iar `x, y` sunt variabile de tip `real` (varianta Pascal) / `float` (varianta C/C++), stabiliți care dintre următoarele secvențe de atribuire sunt incorecte:

- |                           |                           |
|---------------------------|---------------------------|
| a)                        | b)                        |
| x:=20; y:=10; x:=(x+y)/2; | b=2; a=b/2;               |
| x:=8; y:=10; a:=x+y;      | x=8; y=10; a:=x+y;        |
| x:=4; b:=2; y:=x/b;       | x=4; b=2; y=x/b;          |
| b:=2; a:=b/2;             | x:=20; y:=10; x:=(x+y)/2; |
| a:=5; b:=5; x:=a+b;       | a=b=5; x=a+b;             |

43. Care dintre următoarele declarații sunt corecte sintactic?

- |                                  |                            |
|----------------------------------|----------------------------|
| a) <code>const ab=30;</code>     | a) <code>int ab=30;</code> |
| b) <code>var a+b:integer;</code> | b) <code>int a+b;</code>   |
| c) <code>var a1:real;</code>     | c) <code>float a1;</code>  |
| d) <code>var lab:char;</code>    | d) <code>char lab;</code>  |
| e) <code>var m.n:boolean;</code> | e) <code>int m.n;</code>   |

44. Care dintre următoarele expresii logice au valoarea *true* (varianta Pascal)/ 1 (varianta C/C++) pentru:

- |  |  |
|--|--|
| a:=5; b:=3; c:=true; d:=3;                 | a:=5; b:=3; c:=1; d:=3;                            |
| a) <code>(a&lt;b) or c</code>              | a) <code>(a&lt;b)    c</code>                      |
| b) <code>((b=d) and c) or (a&gt;=b)</code> | b) <code>((b==d) &amp;&amp; c)    (a&gt;=b)</code> |
| c) <code>c and (d&gt;b)</code>             | c) <code>c &amp;&amp; (d&gt;b)</code>              |
| d) <code>(a&gt;b) or not (d&lt;a)</code>   | d) <code>(a&gt;b)    !(d&lt;a)</code>              |
| e) <code>(a=b) and c</code>                | e) <code>(a==b) &amp;&amp; c</code>                |

45. Ce valoare are expresia  $E = a/b/c*d - a$ , unde  $a=36$ ,  $b=6$ ,  $c=3$ ,  $d=4$ ?

- a) 36.0      b) 40.0      c) -28.0      d) -38.0      e) - 36.0

46. Determinați care dintre următoarele expresii au valoarea *true* (varianta Pascal)/ 1 (varianta C/C++)

- |  |  |
|--|--|
| a) <code>(3&lt;7) and (2&lt;0) or (6=1+3)</code> | a) <code>(3&lt;7) &amp;&amp; (2&lt;0)    (6==1+3)</code> |
| b) <code>(3&lt;7) or (2&lt;0) or (6=3+3)</code>  | b) <code>(3&lt;7)    (2&lt;0)    (6==3+3)</code>         |
| c) <code>not(2&lt;0) or (6=1+3)</code>           | c) <code>!(2&lt;0)    (6==1+3)</code>                    |

47. Indicați care dintre următoarele expresii sunt incorecte sintactic. Evaluați expresiile corecte.

- |  |  |
|--|--|
| a) <code>3.0 = 5*(10 - 3*3)-2;</code>  | a) <code>3 == 5*(10 - 3*3)-2;</code>   |
| b) <code>25-10 &lt;&gt; 3*5</code>     | b) <code>25-10 != 3*5</code>           |
| c) <code>42 mod 5 &lt; 42 div 5</code> | c) <code>(42 % 5) &lt; (42 / 5)</code> |
| d) <code>10.5 div 2</code>             | d) <code>10.5 % 2</code>               |
| e) <code>4 mod (5&lt;4) div 5</code>   | e) <code>3.14 % (5 &lt; 4) / 5</code>  |
| f) <code>8*3 &lt; 20/10 mod 2</code>   | f) <code>8*3 &lt; 20 % 0.0</code>      |

48. Indicați care dintre expresiile următoare sunt corecte sintactic. Evaluați expresiile corecte.

- |   |   |
|---|---|
| a) <code>3&lt;4 or 5&lt;&gt;6</code>        | a) <code>3&lt;5    5&lt;&gt;6</code>              |
| b) <code>not(true or false)</code>          | b) <code>!(1    0)</code>                         |
| c) <code>not true and false</code>          | c) <code>!1 &amp;&amp; 0</code>                   |
| d) <code>not true or not false</code>       | d) <code>!1    !0</code>                          |
| e) <code>not(18&lt;25)and or(3&gt;0)</code> | e) <code>!(18&lt;25) &amp;&amp;   (3&gt;0)</code> |
| f) <code>12*2 &lt; 2)+18</code>             | f) <code>12*2&lt;2)+18</code>                     |

49. Care dintre declarațiile de variabile următoare sunt corecte?

- |                             |                             |                         |                            |
|-----------------------------|-----------------------------|-------------------------|----------------------------|
| a) <code>var a:byte;</code> | b) <code>var d=char;</code> | a) <code>char a;</code> | b) <code>d=char;</code>    |
| c) <code>var</code>         | d) <code>var</code>         | c) <code>int la;</code> | d) <code>int a;b;c;</code> |
| <code>la:integer;</code>    | <code>a ;b ;c:word;</code>  |                         |                            |

50. Să considerăm următoarele declarații: *var x: byte; y: char;* (varianta Pascal), *unsigned int x; char y;* (varianta C/C++).

Care dintre următoarele afirmații sunt adevărate?

- |   |   |
|---|---|
| a) Variabila x poate avea valoarea 2    | d) Variabila y poate avea valoarea '2'  |
| b) Variabila x poate avea valoarea '12' | f) Variabila y poate avea valoarea '-2' |
| c) Variabila x poate avea valoarea 23   | e) Variabila y poate avea valoarea 256  |

51. Care dintre următoarele expresii care conțin numai operanzi reali sunt corecte sintactic?

- |  |  |
|--|--|
| a) <code>(a&lt;=b) or c</code>             | a) <code>(a&lt;=b)   c</code>          |
| b) <code>(a&lt;&gt;b) and (c&lt;=b)</code> | b) <code>(a!=b) &amp; (c&lt;=b)</code> |
| c) <code>a/c and a/b</code>                | c) <code>a/c &amp; a/b</code>          |
| d) <code>a and b&lt;c+3</code>             | d) <code>a &amp; b&lt;c+3</code>       |
| e) <code>a and b and c</code>              | e) <code>a &amp; b &amp; c</code>      |
| f) <code>(a&lt;&gt;2) and false</code>     | f) <code>(a!=2) &amp; 0</code>         |

52. Se consideră expresia  $(a<b) = ((c+a)>b)$ , descrisă în limbajul Pascal, respectiv  $(a<b) = ((c+a)>b)$  în varianta C/C++. Variabilele întregi care intervin au valorile  $a=2$ ,  $b=30$  și  $c=5$ . Ce valoare se obține în urma evaluării acestei expresii?

- |                              |       |                               |       |
|------------------------------|-------|-------------------------------|-------|
| a) <code>true(Pascal)</code> | b) 32 | c) <code>false(Pascal)</code> | d) 25 |
| 1 (C/C++)                    |       | 0 (C/C++)                     |       |

53. Care dintre variabilele declarate în continuare poate avea valoarea -123?

- |                             |                              |                                |                          |
|-----------------------------|------------------------------|--------------------------------|--------------------------|
| a)                          | b)                           | a)                             | b)                       |
| <code>var nr:byte;</code>   | <code>var b: char;</code>    | <code>unsigned char nr;</code> | <code>unsigned b;</code> |
| c)                          | d)                           | c)                             | d)                       |
| <code>var a:integer;</code> | <code>var a: longint;</code> | <code>int a;</code>            | <code>long a;</code>     |

54. Care va fi valoarea variabilei x după efectuarea secvenței de instrucțiuni:

- |                               |                          |
|-------------------------------|--------------------------|
| <code>x:=3 + 17 div 3;</code> | <code>x=3 + 17/3;</code> |
| <code>y:=x + 1;</code>        | <code>y=x + 1;</code>    |
| <code>x:=y + 1;</code>        | <code>x=y + 1;</code>    |

- a) 8      b) 9      c) 10      d) 11

55. Determinați valoarea expresiei:  $abs(-11.2)+sqrt(trunc(16.23))$ , scrisă în limbajul Pascal, respectiv  $fabs(-11.2)+sqrt(floor(16.23))$  în C/C++.

- a) -9.2      b) 15      c) 14.2      d) 15.2

56. Determinați valoarea expresiei  $\text{trunc}(\text{abs}(-14.2)) \bmod 7$ , scrisă în limbajul Pascal, respectiv  $((\text{int}) \text{fabs}(-14.2)) \% 7$  în C/C++?

- a) 14                      b) 2                      c) 0                      d) 8

57. Determinați valoarea expresiei  $(99 \bmod \text{trunc}(\text{trunc}(8.9)/\text{sqrt}(16)) + 1)$ , scrisă în limbajul Pascal, respectiv  $99 \% (\text{int}) (\text{floor}(8.9)/\text{sqrt}(16)) + 1$  în C/C++?

- a) 2                      b) 1                      c) 0                      d) 3

58. Determinați valoarea expresiei  $\text{sqr}(17 \text{ div } 5 * 2)$ , scrisă în limbajul Pascal, respectiv  $(17/5 * 2) * (17/5 * 2)$  în C/C++.

- a) 1                      b) 36                      c) 1800                      d) 2

59. Determinați care dintre expresiile următoare pot fi atribuite variabilei reale x:

- |                    |                   |                    |                  |
|--------------------|-------------------|--------------------|------------------|
| a) 8 mod sqrt(4)   | b) sqrt(63 mod 2) | a) 8 % sqrt(4)     | b) sqrt(63 % 2)  |
| c) sqr(6 div (-3)) | d) sqrt(sqr(-2))  | c) (6/-3) * (6/-3) | d) sqrt(-2 * -2) |

60. Considerăm că într-un program se lucrează cu variabila reală x a cărei valoare este 10.3.

Ce se va afișa în urma apelului  $\text{write}(x:5.2, \text{abs}(x):6.2, \text{trunc}(x):3)$ , pentru limbajul Pascal, respectiv  $\text{printf}("%5.2f\%6.2f\%3d", x, \text{fabs}(x), (\text{int}) x)$  pentru C/C++?

- a) 10.30-10.30 10    b) 10.3010.3010    c) 10.30 10.30 10    d) 10.30 10.30010

61. Considerând variabila reală x și variabila întreagă a, care va fi secvența de caractere afișate în urma executării secvenței de instrucțiuni:

x:=-4.3; a:=abs(trunc(4)); write(a:2,abs(x):4:2,trunc(a+x):2)	x=-4.3; a=abs(4); printf("%2d%4.2f%2d", a, fabs(x), (int) ceil(a+x));
---	---

- a) 4 4.300                      b) 44.30 0                      c) 4 4.30 0                      d) 44.300

62. Presupunem că variabila întreagă a are valoarea 10 iar variabila întreagă b are valoarea 5. Ce se va afișa în urma executării următoarei secvențe de instrucțiuni?

aux:=1; if a<b then aux:=a; a:=b; b:=aux; write(a,' ',b);	aux=1; if (a<b) aux=a; a=b; b=aux; cout << a<< ' ' << b;
---	--

- a) 5 5                      b) 10 5                      c) 5 1                      d) 10 10

63. Ce se va afișa în urma executării următoarei secvențe de instrucțiuni?

a:=10; b:=a+1; if a<>b then b:=b+1 else a:=a+1; a:=b*a; write(a,' ',b);	a:=10; b:=a+1; if (a!=b) b++; else a++; a:=b*a; cout << a << ' ' << b;
--	---

- a) 121 11                      b) 120 12                      c) 120 11                      d) 121 12

64. Presupunem că asupra variabilelor reale x și y au fost efectuate atribuirile  $x:=10.23$  și  $y:=5.14$ , în Pascal, respectiv  $x=10.23$  și  $y=5.14$ , în C/C++. Ce se va afișa în urma executării următoarei instrucțiuni?

if trunc(x)=trunc(y) then x:=y; else x:=x+y write(trunc(x),' ',trunc(y));	if (floor(x)==floor(y)) x=y; else x+=y; cout << (int)floor(x) << ' ' << (int)floor(y);
--	--

- a) 5 5                      b) 10 15                      c) 15 10                      d) 15 5

65. Știind că variabilele reale x, y, z au valorile  $x=12.3$ ,  $y=-34.2$  și  $z=5.67$ , ce se va afișa în urma executării următoarei instrucțiuni?

if x>y then x:=trunc(x)+1 else if z>y then z:=trunc(z)+1 else y:=trunc(y)+1; writeln(x:3:0,y:3:0,z:3:0);	if (x>y) x=floor(x)+1; else if (z>y) z=floor(z)+1; else y=ceil(y)+1; printf("%3.0f%3.0f%3.0f\n",x,y,z)
--	--

- a) 13-34 6                      b) 12 34 6                      c) 12-35 6                      d) 13 35 5

66. Știind că variabilele caracter x și y au valorile  $x='*'$  și  $y='-'$ , ce se va afișa în urma executării următoarei instrucțiuni?

a:=1; b:=12; c:=4; if a>b then begin if x='*' then a:=a*3 end else if y='-' then b:=b-3 else a:=a+b; write(a,' ',b,' ',c);	a:=1; b:=12; c:=4; if (a>b) { if (x=='*') a*=3; } else { if (y=='-') b-=3; else a+=b; } cout<<a<< ' ' <<b<< ' ' <<c;
---	--

- a) 13 12 4                      b) 3 12 4                      c) 1 9 4                      d) 3 9 4

67. Identificați care dintre următoarele instrucțiuni alternative sunt corecte sintactic:

- |   |                                 |
|---|---------------------------------|
| a) if a:=10 then write(a);                      | a) if a=10 cout<<a;             |
| b) if 1<x<5 then begin<br>x=x+1; write(x); end; | b) if (1<x<5) { x++; cout<<x; } |



c) if (x=3) or (x=5) then write(x);  
 d) if x<10 then begin  
     write(x);  
   end;

68. Care dintre următoarele secvențe de instrucțiuni determină în mod corect maximul dintre trei numere?

a) if (a>b) and (a>c) then max:=a  
     else  
       if (b>a) and (b>c) then max:=b  
       else max:=c;  
 b) if a>b then  
     if a>c then max:=a  
     else max:=c else max:=b  
 c) if a>b then  
     if a>c then max:=a  
     else max:=c  
     else if b>c then max:=b  
     else max:=c;  
 d) if a>b then  
     if b>c then max:=b  
     else max:=c  
     else max:=a;

69. Care dintre următoarele instrucțiuni verifică în mod corect dacă valoarea variabilei x aparține intervalului (a,b)?

a) if (x<=a) or (x>=b) then  
     write('NU Apartine')  
     else write('Apartine')  
 b) if (x>a) and (x<b) then  
     write('Apartine')  
     else write('NU Apartine')  
 c) if (x>a) or (x>b) then  
     write('Apartine')  
     else write('NU Apartine')  
 d) if x>a then  
     if x<b then  
       write('Apartine')  
     else write('NU Apartine')

c) if (x==3 || y==5) cout<<x;  
 d) if (x<10) { cout<<x; }

a) if (a>b && a>c) max=a;  
     else  
       if (b>a && b>c) max=b;  
       else max=c;  
 b) if (a>b)  
     if (a>c) max=a;  
     else max=c; else max=b;  
 c) if (a>b)  
     if (a>c) max=a;  
     else max=c;  
     else if (b>c) max=b;  
     else max=c;  
 d) if (a>b)  
     if (b>c) max=b;  
     else max=c;  
     else max=a;

a) if (x<=a || x>=b)  
     cout << "NU Apartine";  
     else cout << "Apartine";  
 b) if (x>a && x<b)  
     cout << "Apartine";  
     else cout << "NU Apartine";  
 c) if (x>a || x>b)  
     cout << "Apartine";  
     else cout << "NU Apartine";  
 d) if (x>a)  
     if (x<b)  
       cout << "Apartine";  
     else cout << "NU Apartine";

70. Care dintre următoarele instrucțiuni verifică în mod corect dacă valorile variabilelor a, b și c au același semn?

a)  
 if (a>0) and (b>0) and (c>0) then  
     write('Au acelasi semn')  
 else write('NU au acelasi semn')  
 b)  
 if (a\*b>0) and (c\*b>0) then  
     write('Au acelasi semn')  
 else write('NU au acelasi semn')  
 c)  
 if (a>0) and (b>0) and (c>0) or  
     (a<0) and (b<0) and (c<0) then  
     write('Au acelasi semn')  
 else write('NU au acelasi semn')  
 d)  
 if (a\*b<0) or (c\*b<0) or (a\*c<0)  
     then write('NU au acelasi semn')  
 else  
     write('Au acelasi semn')

a)  
 if (a>0 && b>0 && c>0)  
     cout << "Au acelasi semn";  
 else cout << "NU au acelasi semn";  
 b)  
 if (a\*b>0 && c\*b>0)  
     cout << "Au acelasi semn";  
 else cout << "NU au acelasi semn";  
 c)  
 if ((a>0 && b>0 && c>0) ||  
     (a<0 && b<0 && c<0))  
     cout << "Au acelasi semn";  
 else cout << "NU au acelasi semn";  
 d)  
 if (a\*b<0 || c\*b<0 || a\*c<0)  
     cout << "NU au acelasi semn";  
 else  
     cout << "Au acelasi semn";

71. Care dintre următoarele instrucțiuni verifică în mod corect dacă valorile variabilelor a și b sunt consecutive?

a)  
 if (a=b+1) and (a=b-1) then  
     write('sunt consecutive')  
 else  
     write('NU sunt consecutive')  
 b)  
 if (a=b+1) or (a=b-1) then  
     write('sunt consecutive')  
 else  
     write('NU sunt consecutive')  
 c)  
 if (a<>b+1) and (a<>b-1) then  
     write('NU sunt consecutive')  
 else  
     write('sunt consecutive')  
 d)  
 if (a-b=1) and (b-a=1) then  
     write('sunt consecutive')  
 else  
     write('NU sunt consecutive')

a)  
 if (a==b+1 && a==b-1)  
     cout << "sunt consecutive";  
 else  
     cout<<"NU sunt consecutive";  
 b)  
 if (a==b+1 || a==b-1)  
     cout << "sunt consecutive";  
 else  
     cout<<"NU sunt consecutive";  
 c)  
 if (a!=b+1 && a!=b-1)  
     cout<<"NU sunt consecutive";  
 else  
     cout << "sunt consecutive";  
 d)  
 if (a-b==1 && b-a==1)  
     cout << "sunt consecutive";  
 else  
     cout<<"NU sunt consecutive";

### 1.1.2 Teste cu itemi semiobiectivi

1. Se consideră următorul algoritm:

```

1  intreg a, b, c, d;
2  citește a, b;
3  c ← a + b;
4  d ← a * b;
5  dacă c > d atunci
6    c ← d;
7  ■
8  dacă a mod 2 = 0 atunci
9    scrie c, d
10 altfel
11    scrie d, c;
12 ■

```

- a) Ce valori vor fi afișate dacă  $a=34$  și  $b=2$ ? Dar dacă  $a=24$  și  $b=-2$ ?
- b) Dați un exemplu pentru datele de intrare astfel încât algoritmul să afișeze la final două valori egale.
- c) Dați exemplu de valori pentru  $a$  și  $b$  astfel încât să fie afișată o pereche de valori pare ordonate crescător.
- d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

2. Se consideră următorul program pseudocod:

```

1  intreg a, b;
2  real x, y;
3  citește x, y;
4  a ← [x * y];
5  b ← [x / y];
6  dacă a < b atunci
7    a ← b;
8  ■
9  dacă x ≠ [x] atunci
10    scrie a, b
11 altfel
12    scrie b, a;
13 ■

```

- a) Ce valori vor fi afișate dacă  $x=34.50$  și  $y=17.1$ ? Dar dacă  $x=1.5$  și  $y=-2.5$ ?
- b) Dați un exemplu pentru datele de intrare astfel încât algoritmul să afișeze la final aceleași valori care au fost citite la intrare.
- c) Dați exemplu de valori pentru  $x$  și  $y$  astfel încât să fie afișată o pereche de valori ordonate descrescător.
- d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

3. Se consideră următorul algoritm:

```

1  intreg a, b;
2  citește a, b; {a,b>100}
3  dacă a mod 10 < b mod 10 atunci
4    a ← a - a mod 10 + b mod 10
5  ■
6  dacă a div 10 mod 10 > 5 atunci
7    a ← a - a div 10 * 10
8  altfel
9    b ← b - a mod 100;
10 ■
11 scrie a, b;
12

```

- a) Ce valori vor fi afișate dacă  $a=345$  și  $b=238$ ? Dar pentru  $a=7093$  și  $b=211$ ?
- b) Dați un exemplu pentru datele de intrare astfel încât algoritmul să afișeze la final aceleași valori ca cele citite.
- c) Cum trebuie să fie valoarea variabilei  $a$  pentru ca valoarea ei să nu se modifice în urma executării operațiilor din algoritm.
- d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

4. Se consideră următorul program pseudocod:

```

1  intreg s1, s2, m1, m2, g1, g2, s,
2  m, g;
3  citește s1, s2, m1, m2, g1, g2;
4  s ← s1 + s2; m ← m1 + m2;
5  g ← g1 + g2;
6  dacă s > 60 atunci
7    s ← s mod 60; m ← m + 1;
8  ■
9  dacă m > 60 atunci
10   m ← m mod 60; g ← g + 1;
11 ■
12 scrie g, m, s
13
14
15
16
17

```

- a) Ce valori vor fi afișate dacă  $s1=20$ ;  $s2=47$ ;  $m1=20$ ;  $m2=10$ ;  $g1=3$ ;  $g2=38$ ;
- b) Dați un exemplu pentru datele de intrare astfel încât algoritmul să nu efectueze nici una din operațiile existente în cadrul operațiilor de decizie.
- c) Realizați un enunț de problemă a cărei rezolvare este algoritmul prezentat.
- d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

5. Se consideră următorul algoritm:

```

1  intreg a, b, c;
2  citește a, b, c;
3  dacă a*b < 0 atunci
4    scrie 'Exista nr. negativ'
5  altfel
6    dacă b*c < 0 atunci
7      scrie 'Exista nr. negativ'
8    altfel
9      scrie 'NUMERE POZITIVE'
10 ■
11
12
13
14
15
16
17
18

```

- a) Ce mesaj va fi afișat pentru  $a=2$ ,  $b=6$  și  $c=-10$ ? Dar pentru  $a=1$ ,  $b=3$  și  $c=10$ ?
- b) Dați un exemplu de valori pentru datele de intrare astfel încât algoritmul să afișeze un mesaj necorelat cu semnele datelor de intrare.
- c) Rescrieți algoritmul astfel încât să verifice în mod corect dacă toate cele trei valori ale datelor de intrare sunt pozitive.
- d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

6. Se consideră următorul algoritm:

```

1  intreg a, b, c;
2  citește a, b, c;
3  dacă (a + b)/2 = c atunci
4    scrie 'CORECT'
5  altfel
6    dacă (a + c)/2 = b atunci
7      scrie 'CORECT'
8    altfel
9      dacă (c+b)/2 = a atunci
10        scrie 'CORECT'
11      altfel
12        scrie 'INCORECT';

```

- a) Ce valoare va fi afișată dacă  $a=3$ ,  $b=8$  și  $c=13$ ? Dar pentru  $a=5$ ,  $b=10$  și  $c=0$ ?
- b) Dați un exemplu pentru datele de intrare astfel încât algoritmul să afișeze mesajul 'INCORECT'.
- c) Rescrieți algoritmul, folosindu-vă de operațiile logice, astfel încât să conțină o singură operație de decizie.
- d) Realizați programul în limbajul de

```

13
14
15

```

7. Se consideră următorul algoritmul:

```

1  intreg a, b, c;
2  citește a, b, c;
3  dacă a > b atunci
4      a ↔ b
5  ■
6  dacă b > c atunci
7      c ↔ b
8  altfel
9      dacă a > c atunci
10         a ↔ c;
11 ■
12 ■
13 scrie a, b, c;

```

8. Se consideră următorul algoritmul:

```

1  intreg a, b, c;
2  citește a, b, c; {a,b,c>0}
3  dacă (a+b)<c atunci
4      scrie 'NU'
5  altfel
6      dacă (c+b) < a atunci
7          scrie 'NU'
8      altfel
9          dacă (c+a) < b atunci
10             scrie 'NU'
11             altfel scrie 'Corect'
12 ■
13 ■
14 ■

```

programare studiat Pascal/C/C++.

e) Realizați un enunț de problemă a cărei rezolvare este algoritmul prezentat.

a) Ce valori vor fi afișate dacă  $a=3$ ,  $b=38$  și  $c=17$ ? Dar pentru  $a=73$ ,  $b=15$  și  $c=46$ ?

b) Dați un exemplu pentru datele de intrare astfel încât algoritmul să afișeze valori ce nu sunt ordonate crescător.

c) Cum trebuie modificat algoritmul pentru ca valorile variabilelor  $a$ ,  $b$ ,  $c$  să fie ordonate crescător?

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

a) Ce mesaj va fi afișat pentru  $a=2$ ,  $b=6$  și  $c=10$ ? Dar pentru  $a=3$ ,  $b=4$  și  $c=5$ ?

b) Realizați un enunț de problemă a cărei rezolvare este algoritmul prezentat.

c) Realizați un algoritm echivalent, folosindu-vă de operațiile logice, care să conțină o singură operație de decizie.

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

9. Se consideră următorul algoritmul:

```

1  intreg a;
2  citește a;
3  dacă a mod 100 < 50 atunci
4      a ← a - a mod 100
5  altfel
6      a ← a + 100 - a mod 100
7  ■
8  scrie a; stop.

```

10. Se consideră următorul algoritmul:

```

1  real x;
2  citește x;
3  x ← x*10
4  dacă [x] mod 10 ≠ 0 atunci
5      x ← [x]/10
6  altfel
7      x ← x*10;
8      dacă [x] mod 10 ≠ 0 atunci
9          x ← [x]/100
10     altfel
11         x ← x/1000
12 ■
13 ■
14 scrie x; stop.

```

11. Se consideră următorul algoritmul:

```

1  real a, b, c, x1, x2, delta;
2  citește a, b, c;
3  dacă a=0 atunci
4      scrie "Ecuație de grad I"
5  altfel
6      delta ← b*b - 4*a*c
7      dacă delta>=0 atunci
8          x1 ← (-b - √delta)/(2*a)
9          x2 ← (-b + √delta)/(2*a)
10         scrie x1, x2
11     altfel
12         scrie "Nu sunt sol reale"
13 ■
14 ■

```

a) Ce valori se va afișa pentru  $a=2345$ ? Dar pentru  $a=70189$ ?

b) Realizați un enunț de problemă a cărei rezolvare este algoritmul prezentat.

c) Realizați programul în limbajul de programare studiat Pascal/C/C++.

a) Ce valori se va afișa pentru  $x=12.345$ ? Dar pentru  $x=12.034$ ?

b) Dați exemplu de valoare pentru  $x$  astfel încât la finalul algoritmului partea întreagă a lui să fie diferită față de cea avută la intrare.

c) Dați exemplu de valoare pentru  $x$  astfel încât, la final, să fie afișată o valoare cu parte fracționară egală cu 0.00.

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

a) Ce mesaj va fi afișat pentru  $a=1$ ,  $b=2$ ,  $c=1$ ? Dar pentru  $a=4$ ,  $b=-12$ ,  $c=9$ ?

b) Dați exemplu de valori naturale citite pentru  $a$ ,  $b$ ,  $c$  astfel încât să se tiparească mesajul 'Nu sunt sol reale'.

c) Dați exemplu de două seturi de valori de intrare pentru care una din valorile afișate să fie 0.

d) Identificați care din datele de intrare pot fi declarate ca date de tip întreg.

e) Realizați programul în limbajul de programare studiat Pascal/C/C++.

### 1.1.3 Probleme rezolvate

1. Considerând cunoscute trei valori reale, verificați dacă ele pot reprezenta lungimile laturilor unui triunghi, iar în caz afirmativ determinați tipul acestuia: isoscel, echilateral, dreptunghic sau oarecare.

**Soluție:** Algoritmul testează toate cazurile după definițiile triunghiului echilateral, isoscel, respectiv dreptunghic.

```

1 var a,b,c:real;
2 begin
3   readln(a,b,c);
4   if (a<0)or(b<0)or(c<0) then
5     writeln('Lungimi negative')
6   else
7     if(a>=b+c)or(b>=a+c)or(c>=a+b) then
8       writeln('Nu este triunghi')
9     else
10      if (a=b)and(a=c)and(b=c) then
11        writeln('Echilateral')
12      else
13        if (a=b)or(a=c)or(b=c) then
14          writeln('Isoscel')
15        else
16          if (sqr(a)+sqr(b)=sqr(c))or
17            (sqr(b)+sqr(c)=sqr(a))or
18            (sqr(a)+sqr(c)=sqr(b)) then
19            writeln('Dreptunghic')
20          else
21            writeln('Oarecare');
22        end.
23
#include <iostream.h>
float a,b,c;
void main(){
  cin >> a >> b >> c;
  if (a<0 || b<0 || c<0)
    cout <<"Lungimi negative\n";
  else
    if (a>=b+c || b>=a+c || c>=a+b)
      cout <<"Nu este triunghi\n";
    else
      if (a==b && a==c && b==c)
        cout <<"Echilateral\n";
      else
        if (a==b || a==c || b==c)
          cout <<"Isoscel\n";
        else
          if (a*a+b*b==c*c ||
              b*b+c*c==a*a ||
              a*a+c*c==b*b)
            cout <<"Dreptunghic\n";
          else
            cout <<"Oarecare\n";
}
```

2. Se consideră o fracție a cărei numitor este un număr prim. Să se verifice dacă este ireductibilă și subunitară.

**Soluție:** Frația este ireductibilă dacă numărătorul nu este un multiplu al numitorului. Variabila  $a$  va prelua valoarea numărătorului.

```

1 var a,b:integer;
2 begin
3   readln(a,b);
4   if (a mod b=0) then
5     write('Reducibila,')
6   else
7     write('Ireductibila,')
8   if (a<b) then
9     write('Subunitara')
10  else
#include <iostream.h>
int a,b;
void main() {
  cin >> a >> b ;
  if (a%b==0)
    cout << "Reducibila";
  else
    cout << "Ireductibila";
  if (a<b)
    cout << "Subunitara";
}
```

```

11 write('Supraunitara')
12 end.
else
  cout << "Supraunitara";
}
```

3. Fie  $x$  un număr natural de cel mult 9 cifre. Să se determine ultima cifră a puteri  $2^x$ .

**Soluție:** Ultima cifră a puterilor lui 2 se repetă periodic (din 4 în 4).

```

1 var x:longint;
2 begin
3   readln(x);
4   if x=0 then write(1)
5   else
6     if x mod 4 =1 then write(2)
7     else
8       if x mod 4=2 then write(4)
9       else
10        if x mod 4=3 then write(8)
11        else write(6)
12   end.
#include <iostream.h>
long x ;
void main() {
  cin >> x;
  if (x==0) cout << "0";
  else
    if (x%4==1) cout << "2";
    else
      if (x%4==2) cout << "4";
      else
        if (x%4==3) cout << "8";
        else cout << "6";
}
```

4. Se consideră două triunghiuri în plan, identificate prin coordonatele vârfurilor lor. Să se realizeze un program care verifică dacă cele două triunghiuri sunt asemenea.

**Soluție:** Două triunghiuri sunt asemenea dacă lungimile laturilor determină rapoarte egale(cf. teoremei fundamentale a asemănării). Fie două puncte în plan  $A(x_a, y_a)$  și  $B(x_b, y_b)$ . Distanța dintre cele două puncte este:  $\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$ . Cu ajutorul acestei formule se determină lungimile laturilor celor două triunghiuri, verificându-se apoi dacă rapoartele care se formează sunt egale.

```

1 var
2 x1,y1,x2,y2,x3,y3,x4,y4:real;
3 a,b,c,d,e,f,x5,y5,x6,y6:real;
4 begin
5   read(x1,y1,x2,y2);
6   read(x3,y3,x4,y4);
7   read(x5,y5,x6,y6);
8   a:=(x1-x2)*(x1-x2) +
9     (y1-y2)*(y1-y2);
10  b:=(x1-x3)*(x1-x3) +
11    (y1-y3)*(y1-y3);
12  c:=(x2-x3)*(x2-x3) +
13    (y2-y3)*(y2-y3);
14  d:=(x4-x5)*(x4-x5) +
15    (y4-y5)*(y4-y5);
16  e:=(x4-x6)*(x4-x6) +
17    (y4-y6)*(y4-y6);
18  f:=(x5-x6)*(x5-x6) +
19    (y5-y6)*(y5-y6);
#include <iostream.h>
float x1,y1,x2,y2,x3,y3,x4,y4,
x5,y5,x6,y6;
float a,b,c,d,e,f;
void main() {
  cin >> x1 >> y1 >> x2 >> y2;
  cin >> x3 >> y3 >> x4 >> y4;
  cin >> x5 >> y5 >> x6 >> y6;
  a=(x1-x2)*(x1-x2) +
    (y1-y2)*(y1-y2);
  b=(x1-x3)*(x1-x3) +
    (y1-y3)*(y1-y3);
  c=(x2-x3)*(x2-x3) +
    (y2-y3)*(y2-y3);
  d=(x4-x5)*(x4-x5) +
    (y4-y5)*(y4-y5);
  e=(x4-x6)*(x4-x6) +
    (y4-y6)*(y4-y6);
  f=(x5-x6)*(x5-x6) +
    (y5-y6)*(y5-y6);
}
```

```

20 if (a/d=b/e)and(a/d=c/f)
21 then
22     write('DA')
23 else
24     write('NU');
24 end.

```

```

(y5-y6)*(y5-y6);
if (a/d==b/e && a/d==c/f)
    cout << "DA";
else
    cout << "NU";
}

```

5. Se consideră două puncte în plan, exprimate prin perechi de coordonate  $(x,y)$ . Ele reprezintă centrele a două cercuri de rază  $R1$ , respectiv  $R2$ . Să se verifice dacă cele două cercuri sunt tangente interne, tangente externe, secante sau exterioare.

**Soluție:** Pentru a determina poziția celor două cercuri vom calcula distanța dintre cele două centre. Notând cu  $d$  această distanță, atunci:

- cercurile sunt exterioare dacă  $d > R1 + R2$
- cercurile sunt tangente externe dacă  $d = R1 + R2$
- cercurile sunt tangente interne dacă  $d = |R1 - R2|$
- cercurile sunt secante dacă  $d < |R1 - R2|$  și  $d < R1 + R2$
- un cerc este interior celuilalt dacă  $d < |R1 - R2|$

```

1 var x1,y1,r1,x2,y2,r2,d:real;
2 begin
3     read(x1,y1,r1);
4     read(x2,y2,r2);
5     d:=sqrt((x1-x2)*(x1-x2)+
6         (y1-y2)*(y1-y2));
7     if (d>r1+r2) then
8         write('Exterioare')
9     else
10    if (d=r1+r2) then
11        write('Tangente ext')
12    else
13    if (d<r1+r2)and(d>abs(r1-r2))
14        then
15            write('Secante')
16    else
17        if (d=abs(r1-r2)) then
18            write('Tangente int')
19        else
20            if (d<abs(r1-r2)) then
21                write('Interioare')
22    end.

```

```

#include <iostream.h>
#include <math.h>
float x1,y1,r1,x2,y2,r2,d;
void main() {
    cin >> x1 >> y1 >> r1;
    cin >> x2 >> y2 >> r2;
    d=sqrt((x1-x2)*(x1-x2)+
        (y1-y2)*(y1-y2));
    if (d>r1+r2)
        cout << "Exterioare";
    else
        if (d==r1+r2)
            cout << "Tangente ext";
        else
            if (d<r1+r2 &&d>fabs(r1-r2))
                cout << "Secante";
            else
                if (d=fabs(r1-r2))
                    cout << "Tangente int";
                else
                    if (d<fabs(r1-r2))
                        cout << "Interioare";
}

```

6. Se consideră patru puncte în plan  $A, B, C, D$ , exprimate prin perechea de coordonate  $(x,y)$  ce formează un patrulater convex. Să se verifice dacă cele patru puncte formează un paralelogram. Coordonatele punctelor sunt introduse în ordinea  $(x_A, y_A) (x_B, y_B) (x_C, y_C) (x_D, y_D)$ .

**Soluție:** Patru puncte formează un paralelogram dacă diagonalele se înjumătățesc în același punct. Vom verifica dacă coordonatele mijloacelor celor două diagonale sunt aceleași.

```

1 var
2 x1,y1,x2,y2,x3,y3,x4,y4:real;
3 begin
4     read(x1,y1,x2,y2);
5     read(x3,y3,x4,y4);
6     if (x1+x3=x2+x4)and
7         (y1+y3=y2+y4) then
8         write('DA')
9     else
10        write('NU');
11    end.

```

```

#include <iostream.h>
float x1,y1,x2,y2,x3,y3,x4,y4;
void main() {
    cin >> x1 >> y1 >> x2 >> y2;
    cin >> x3 >> y3 >> x4 >> y4;
    if (x1+x3==x2+x4 &&
        y1+y3==y2+y4)
        cout << "DA";
    else
        cout << "NU";
}

```

7. Realizați un program care verifică dacă un punct  $X$  din plan se află în interiorul unui triunghi, pe laturile acestuia sau este exterior lui. Se cunosc coordonatele punctului  $A$  și coordonatele vârfurilor triunghiului.

**Soluție:** Punctul  $X$  este interior triunghiului  $ABC$  dacă suma ariilor triunghiurilor  $AXB, XAC$  și  $XBC$  este egală cu aria triunghiului  $ABC$ . Dacă una din ariile triunghiurilor care îl au ca vârf pe  $X$  este egală cu 0 atunci punctul  $X$  se află pe laturile triunghiului  $ABC$ .

```

1 var
2 x1,y1,x2,y2,x3,y3,x,y:real;
3 a,a1,a2,a3:real;
4 begin
5     read(x1,y1,x2,y2);
6     read(x3,y3,x,y);
7     a:=abs(x1*(y2-y3)-y1*
8         (x2-x3)+x2*y3-x3*y2)*0.5;
9     a1:=abs(x1*(y2-y)-y1*
10        (x2-x)+x2*y-x*y2)*0.5;
11    a2:=abs(x1*(y-y3)-y1*
12        (x-x3)+x*y3-x3*y)*0.5;
13    a3:=abs(x*(y2-y3)-y*
14        (x2-x3)+x2*y3-x3*y2)*0.5;
15    if (a1=0)or(a2=0)or(a3=0)
16        then write('Pe laturi')
17    else
18        if (a1+a2+a3=a) then
19            write('Interior')
20        else write('Exterior')
21    end.

```

```

#include <iostream.h>
#include <math.h>
float x1,y1,x2,y2,x3,y3,x,y;
float a,a1,a2,a3;
void main() {
    cin >> x1 >> y1 >> x2 >> y2;
    cin >> x3 >> y3 >> x >> y;
    a=fabs(x1*(y2-y3)-y1*
        (x2-x3)+x2*y3-x3*y2)*0.5;
    a1=fabs(x1*(y2-y)-y1*
        (x2-x)+x2*y-x*y2)*0.5;
    a2=fabs(x1*(y-y3)-y1*
        (x-x3)+x*y3-x3*y)*0.5;
    a3=fabs(x*(y2-y3)-y*
        (x2-x3)+x2*y3-x3*y2)*0.5;
    if (a1==0 || a2==0 || a3==0)
        cout << "Pe laturi";
    else
        if (a1+a2+a3 == a)
            cout << "Interior";
        else cout << "Exterior";
}

```

8. Se consideră trei puncte în plan, exprimate prin perechi de coordonate  $(x,y)$ . Să se verifice dacă un punct  $A$  reprezintă centrul de greutate al triunghiului format de cele trei puncte.

**Soluție:** Coordonatele centrului de greutate al unui triunghi se obțin ca medie aritmetică a celor trei coordonate corespunzătoare vârfurilor triunghiului. Demonstrația pleacă de la faptul că acesta se găsește pe orice mediană la  $2/3$  de coordonatele vârfului și la  $1/3$  de bază.

```

1 var x1,y1,x2,y2,x3,y3,xa,ya:real;
2   x1,y1,x2,y2,x3,y3,xa,ya:real;
3 begin
4   read(x1,y1,x2,y2);
5   read(x3,y3,xa,ya);
6   if ((x1+x2+x3)/3.0=xa)and
7     ((y1+y2+y3)/3.0=ya)then
8     write('Centru de greutate')
9   else
10    write('NU')
11 end.

```

```

#include <iostream.h>
float x1,y1,x2,y2,x3,y3,xa,ya;
void main() {
  cin >> x1 >> y1 >> x2 >> y2;
  cin >> x3 >> y3 >> xa >> ya;
  if ((x1+x2+x3)/3.0 == xa &&
      (y1+y2+y3)/3.0 == ya)
    cout<<"Centru de greutate";
  else
    cout << "NU";
}

```

9. Se consideră patru puncte în plan, exprimate prin perechea de coordonate  $(x,y)$ . Să se verifice dacă cele patru puncte formează un dreptunghi.

**Soluție:** Patru puncte formează un dreptunghi dacă diagonalele sunt egale și se înjumătățesc în același punct.

```

1 var x1,y1,x2,y2,x3,y3:real;
2   d1,d2,x4,y4:real;
3 begin
4   read(x1,y1,x2,y2);
5   read(x3,y3,x4,y4);
6   d1:=(x1-x3)*(x1-x3)+
7     (y1-y3)*(y1-y3);
8   d2:=(x2-x4)*(x2-x4)+
9     (y2-y4)*(y2-y4);
10  if (x1+x3=x2+x4) and
11    (y1+y3=y2+y4)and
12    (d1=d2) then
13    write('DA')
14  else
15    write('NU')
16 end.

```

```

#include <iostream.h>
float x1,y1,x2,y2,x3,y3,x4,y4;
float d1,d2;
void main() {
  cin >> x1 >> y1 >> x2 >> y2;
  cin >> x3 >> y3 >> x4 >> y4;
  d1=(x1-x3)*(x1-x3)+
    (y1-y3)*(y1-y3);
  d2=(x2-x4)*(x2-x4)+
    (y2-y4)*(y2-y4);
  if (x1+x3==x2+x4 &&
      y1+y3==y2+y4 && d1==d2)
    cout << "DA";
  else
    cout << "NU";
}

```

10. Se consideră două dreptunghiuri, cu laturile paralele cu axele, în plan ale căror vârfuri sunt exprimate prin perechi de coordonate  $(x,y)$ . Să se determine aria suprafeței comune ale celor două dreptunghiuri. Coordonatele punctelor unui dreptunghi sunt introduse în ordinea  $(x_A,y_A)$ ,  $(x_B,y_B)$ ,  $(x_C,y_C)$ ,  $(x_D,y_D)$ .

**Soluție:** Calculăm lungimea și lățimea suprafeței comune, apoi determinăm aria acestei suprafețe.

```

1 var x1,y1,x2,y2,x3,y3,x4,y4,
2   x5,y5,x6,y6,x7,y7,x8,y8,
3   vx,vy:real;
4 begin
5   read(x1,y1,x2,y2);
6   read(x3,y3,x4,y4);
7   read(x5,y5,x6,y6);
8   read(x7,y7,x8,y8);
9   if (x1<=x5)and(x5<=x2)and
10     (x1<=x6)and(x6<=x2)then
11     vx:=x6-x5
12   else
13     if (x5<=x1)and(x1<=x6)and
14       (x5<=x2)and(x2<=x6)then
15       vx:=x2-x1
16   else
17     if (x1<=x5)and(x5<=x2)then
18       vx:=x2-x5
19   else
20     if (x1<=x6)and(x6<=x2)then
21       vx:=x6-x1;
22
23   if (y1<=y5)and(y5<=y2)and
24     (y1<=y6)and(y6<=y2)then
25     vy:=y6-y5
26   else
27     if (y5<=y1)and(y1<=y6)and
28       (y5<=y2)and(y2<=y6)then
29       vy:=y2-y1
30   else
31     if (y1<=y5)and(y5<=y2)then
32       vy:=y2-y5
33   else
34     if (y1<=y6)and(y6<=y2)then
35       vy:=y6-y1;
36   write(vx*vy);
37 end.

```

```

#include <iostream.h>
float x1,y1,x2,y2,x3,y3,x4,y4,
x5,y5,x6,y6,x7,y7,x8,y8;
float vx,vy;
void main() {
  cin >> x1 >> y1 >> x2 >> y2;
  cin >> x3 >> y3 >> x4 >> y4;
  cin >> x5 >> y5 >> x6 >> y6;
  cin >> x7 >> y7 >> x8 >> y8;
  if (x1<=x5 && x5<=x2 &&
      x1<=x6 && x6<=x2)
    vx=x6-x5;
  else
    if (x5<=x1 && x1<=x6 &&
        x5<=x2 && x2<=x6)
      vx=x2-x1;
  else
    if (x1<=x5 && x5<=x2)
      vx=x2-x5;
  else
    if (x1<=x6 && x6<=x2)
      vx=x6-x1;

  if (y1<=y5 && y5<=y2 &&
      y1<=y6 && y6<=y2)
    vy=y6-y5;
  else
    if (y5<=y1 && y1<=y6 &&
        y5<=y2 && y2<=y6)
      vy=y2-y1;
  else
    if (y1<=y5 && y5<=y2)
      vy=y2-y5;
  else
    if (y1<=y6 && y6<=y2)
      vy=y6-y1;
  cout << vx*vy;
}

```

### 1.1.4 Probleme propuse

1. Realizați un algoritm pentru calculul expresiilor următoare:  $A=2+x-y$ ;  $B=x*A-2+y$ ;  $C=A-2*B+x$ .
2. Realizați un algoritm pentru determinarea perimetrului și ariei unui triunghi căruia i se cunosc lungimile laturilor.
3. Realizați un algoritm pentru rezolvarea în mulțimea numerelor reale a ecuației de gradul 1 ( $a*x + b = 0$ ).

4. Realizați un algoritm pentru rezolvarea în mulțimea numerelor reale a sistemului de ecuații:

$$\begin{cases} a \cdot x + b \cdot y = 0 \\ x + c \cdot y = 1 \end{cases}$$

5. Inflația produce devalorizarea monedei naționale. Cât a costat un caiet anul trecut, dacă inflația a fost de 50% pe an? Alcătuiți algoritmul de rezolvare a acestei probleme știind care este prețul actual al caietului.

6. Andrei a depus la o bancă o sumă de bani. Știm că în fiecare lună el primește o dobândă de 30% din valoarea inițial depusă. Care va fi valoarea deținută la C.E.C după 3 luni dacă se cunoaște valoarea depusă inițial? Alcătuiți algoritmul de rezolvare al acestei probleme.

7. Maria a depus la bancă o sumă de bani. Se cunoaște că pe lună ea are o dobândă de 30% din valoarea avută la acel moment. Care va fi valoarea deținută la C.E.C după 3 luni cunoscând suma inițial depusă și faptul că Maria nu a ridicat niciodată dobânda, ea adăugându-se în fiecare lună la suma inițială? Alcătuiți algoritmul de rezolvare al acestei probleme.

8. Realizați un algoritm care citind de la tastatură trei numere reale calculează suma celor pozitive.

9. Realizați câte un algoritm pentru calculul valorilor următoarelor expresii:

$$A = \begin{cases} 3 \cdot x - 1 & \text{daca } x < 0 \\ 2 - x & \text{daca } x \geq 0 \end{cases}$$

$$C = \begin{cases} 2 - x & \text{daca } x < 0 \\ \max\{x^2, 1/x\} & \text{daca } x \geq 0 \end{cases}$$

$$B = \begin{cases} 5 \cdot x & \text{daca } x < 10 \\ 2 - x & \text{daca } 0 \leq x \leq 10 \\ 3 + x & \text{daca } x > 10 \end{cases}$$

10. Realizați un algoritm care verifică dacă trei numere întregi sunt pitagorice.

11. Realizați un program care va determina media semestrială la o disciplină având la dispoziție cele trei note din oral.

12. Realizați un program care va determina media semestrială la o disciplină având la dispoziție cele trei note din oral și nota din teză.

13. Realizați un program care va determina lungimea ipotenuzei unui triunghi dreptunghic, cărui i se cunosc lungimile catetelor.

14. Realizați un program care determină perimetrul și aria unui pătrat, cunoscându-se lungimea laturii acestuia.

15. Realizați un program care determină un procent  $p$  din salariul unui muncitor. Se vor citi de la tastatură atât salariul cât procentul dorit.

*Exemplu:* Pentru salariul  $s=3000000$  și procentul  $p=15$  se va afișa 450000.

16. Cunoscându-se suma de bani avută de Ionel înainte de a cumpăra trei cadouri, și valoarea fiecăruia, realizați un program care să afișeze suma rămasă după cumpărături.

*Exemplu:* Pentru suma inițială  $s=3000000$  și cadourile de valoare 100000, 200000 și 300000 se va afișa 2400000.

17. O broscuță face în fiecare minut câte un salt. Lungimea primului salt este  $p$  (valoare citită de la tastatură) după care, fiecare salt făcut are lungimea dublă față de lungimea saltului anterior făcut. Realizați un program care afișează distanța totală parcursă de broscuță în cinci salturi.

*Exemplu:* pentru  $p=2$  se va afișa 62 ( $2+4+8+16+32$ )

18. Considerăm un număr  $n > 100$ . Realizați un program care afișează ultimele două cifre ale lui și suma acestora.

*Exemplu:*

Pentru  $n=10234$  se va afișa 34 7.

19. Considerăm un număr  $n$  de patru cifre. Realizați un program care afișează primele două cifre ale lui și produsul acestora.

*Exemplu:* Pentru  $n=1234$  se va afișa 12 2.

20. Considerăm un număr  $n$  de trei cifre. Realizați un program care afișează pe câte o linie fiecare cifră a acestuia.

*Exemplu:*

Pentru  $n=123$  se va afișa :

1  
2  
3

21. Realizați un program care, citind de la tastatură trei numere reale, calculează suma celor pozitive.

22. Realizați câte un algoritm pentru calculul valorilor următoarelor expresii

$$A = \begin{cases} \min(x \cdot y, x + z) & \text{ptr } x \neq y \\ 2 \cdot z & \text{altfel} \end{cases}$$

$$B = \begin{cases} \min(x \cdot y, x + z) & \text{ptr } x \neq z \\ \max(2 \cdot z, x - y) & \text{altfel} \end{cases}$$

23. Realizați un program care afișează în ordine crescătoare cifrele unităților și zecilor unui număr natural  $n$  citit.

*Exemplu:* Pentru  $n=1243$  se va afișa 3 4

24. Realizați un program care verifică dacă un număr este impar și afișează în caz afirmativ suma dintre cifra zecilor și cifra unităților. În situația când numărul citit este par se va afișa mesajul "Numar par"

*Exemplu:*

Pentru  $n=1243$  se va afișa 7.

25. Realizați un program care determină suma a două unghiuri exprimate în grade, minute și secunde.

*Exemplu:*

Pentru primul unghi de măsură: 37 10 22 (grade, minute secunde), pentru al doilea unghi de măsură: 10 52 15, se va afișa 48 2 37.

26. Realizați un program care citind de la tastatură o valoare  $x$  ( $x < 365$ ), afișează pe ecran în ce lună a anului se află ziua cu numărul  $x$  a anului.

*Exemplu:*

Pentru  $x=33$  se va afișa "februarie"

27. Se citește de la tastatură o valoare  $x$  naturală ( $x > 10$ ). Dacă valoarea citită are cifra unităților egală cu cifra zecilor se vor afișa primele două numere naturale mai mari decât valoarea  $x$ . În caz contrar se va afișa cifra maximă dintre cifra zecilor și a unităților.

*Exemplu:*

Pentru  $n=3455$  se va afișa 3456 3457

Pentru  $n=3485$  se va afișa 8

28. Se citește o valoare  $x$  număr natural. Să se realizeze un program care va afișa cele mai apropiate două numere pare de numărul  $x$ .

*Exemplu:* Pentru  $x=14$  se va afișa 12 și 16

Pentru  $x=15$  se va afișa 14 și 16

29. Se citesc două numere naturale. Să se afișeze valoarea fracției subunitare care se poate forma cu cele două numere. Valoarea va fi afișată cu 3 zecimale.

*Exemplu:*

Pentru  $x=2$  și  $y=4$  se va afișa 0.500.

30. Realizați un program care afișează semnul majoritar a 5 valori citite de la tastatură și afișează un mesaj corespunzător.

*Exemplu:*

Pentru valorile 2, -3, -14.2, 2, -4.3 se va afișa mesajul: "Majoritatea negative"

## 1.2. Structuri repetitive – Instrucțiuni repetitive

### 1.2.1 Teste cu alegere multiplă și duală

1. Care dintre următoarele instrucțiuni efectuează 5 iterații:

a) 

```
pentru d ← 0,5 executa
  nr ← nr+1
```

b) 

```
pentru j ← -1,3 executa
  nr ← nr+1
```

c) 

```
pentru d ← -1, 4 executa
  nr ← nr+1
```

d) 

```
pentru j ← 7,3 -1 executa
  nr ← nr+1
```

2. Care dintre următoarele instrucțiuni permite afișarea, în ordine, a valorilor 10, 12, 14, 16, 18, 20?

a) 

```
pentru d ← 8,18 executa
  scrie (d + 2);
```

b) 

```
pentru j ← 10,20 executa
  daca j mod 2=0 atunci
    scrie j;
```

c) 

```
pentru d ← 0,5 executa
  scrie (10 + d*2);
```

d) 

```
pentru j ← 0,10 executa
  scrie (j*2 + 10);
```

3. Ce se va afișa pe ecran în urma executării instrucțiunilor ?

```
s←0;
pentru j ← 11,15 executa
  daca j mod 2=0 atunci s ← s - j mod 2
  altfel s ← s + j mod 2
scrie s;
stop.
```

a) 3

b) 4

c) -4

d) -3

4. Care este valoarea finală pe care o poate lua contorul pentru ca în urma executării instrucțiunii de mai jos să se afișeze valoarea 4 ?

```
nr←0;
pentru i ← 10, ? , -1 executa
  daca i mod 2=0 atunci
    nr ← nr + 1
```



```
scrie nr;
stop.
```

- a) 1                      b) 2                      c) 3                      d) 4

5. Care dintre variantele următoare reprezintă un algoritm care verifică în mod corect dacă un număr este prim?

a) citește  $n$  ( $n > 1$ )  
 $nr \leftarrow 0$   
 pentru  $d \leftarrow 1, n$  executa  
   daca  $n \bmod d = 0$  atunci  
      $nr \leftarrow nr + 1$   
 ■  
 daca  $nr = 0$  atunci  
   scrie "Prim"  
 altfel  
   scrie "Nu este prim"

b) citește  $n$  ( $n > 1$ )  
 $nr \leftarrow 0$   
 pentru  $d \leftarrow 2, n-1$  executa  
   daca  $n \bmod d = 0$  atunci  
      $nr \leftarrow nr + 1$   
 ■  
 daca  $nr = 0$  atunci  
   scrie "Prim"  
 altfel  
   scrie "Nu este prim"

c) citește  $n$  ( $n > 1$ )  
 $nr \leftarrow 0$   
 pentru  $d \leftarrow 1, n$  executa  
   daca  $n \bmod d = 0$  atunci  
      $nr \leftarrow nr + 1$   
 ■  
 daca  $nr = 2$  atunci  
   scrie "Prim"  
 altfel  
   scrie "Nu este prim"

d) citește  $n$  ( $n > 1$ )  
 $nr \leftarrow 0$   
 pentru  $d \leftarrow 2, n-1$  executa  
   daca  $n \bmod d = 0$  atunci  
      $nr \leftarrow nr + 1$   
 ■  
 daca  $nr = 0$  atunci  
   scrie "Prim"  
 altfel  
   scrie "Nu este prim"

6. Care dintre următoarele instrucțiuni conduce la afișarea pe ecran a valorii 5?

a)  
 $nr \leftarrow 0; i \leftarrow 0;$   
 cat timp  $nr < 10$  executa  
   daca  $nr \bmod 10 = 0$  atunci  
      $nr \leftarrow nr + 5;$   
      $i \leftarrow i + 1;$   
 ■  
 $nr \leftarrow nr + 5;$   
 ■  
 scrie  $i.$

b)  
 $nr \leftarrow 1234; i \leftarrow 1;$   
 cat timp  $nr > 9$  executa  
    $i \leftarrow i + 1;$   
    $nr \leftarrow nr \div 100;$   
 ■  
 scrie  $i.$

c)  
 $nr \leftarrow 0; i \leftarrow 0;$   
 cat timp  $nr < 45$  executa  
   daca  $nr \bmod 10 = 0$  atunci  
      $nr \leftarrow nr + 5;$   
      $i \leftarrow i + 1;$   
 ■  
 $nr \leftarrow nr + 5;$   
 ■  
 scrie  $i.$

d)  
 $nr \leftarrow 1357; i \leftarrow 1;$   
 cat timp  $nr > 0$  executa  
    $i \leftarrow i + 1;$   
    $nr \leftarrow nr \div 10;$   
 ■  
 scrie  $i.$

7. Care dintre următoarele secvențe de instrucțiuni permit afișarea valorii 731?

a)  
 $nr \leftarrow 1327; i \leftarrow 0;$   
 cat timp  $nr < 0$  executa  
   daca  $nr \bmod 2 = 1$  atunci  
      $i \leftarrow i * 10 + nr \bmod 10;$   
 ■  
 $nr \leftarrow nr \div 10;$   
 ■  
 scrie  $i.$

b)  
 $nr \leftarrow 731;$   
 cat timp  $nr > 0$  executa  
   scrie  $nr \bmod 10$   
    $nr \leftarrow nr \div 100;$   
 ■

c)  
 $nr \leftarrow 137; i \leftarrow 0;$   
 cat timp  $nr > 0$  executa  
    $i \leftarrow i * 10 + nr \bmod 10;$   
    $nr \leftarrow nr \div 10;$   
 ■  
 scrie  $i$

d)  
 $nr \leftarrow 12307; i \leftarrow 0;$   
 cat timp  $nr > 0$  executa  
    $i \leftarrow i * 10 + nr \bmod 10;$   
    $nr \leftarrow nr \div 100;$   
 ■  
 scrie  $i.$

8. Se consideră următorul algoritm descris în pseudocod. Ce se va afișa în urma executării instrucțiunilor (operatorul % reprezintă restul la împărțirea întreagă):

```
intreg a, b;
a ← 10; b ← 2;
cat timp a < 15 executa
  daca a % b = 0 atunci scrie "*"
  a ← a + 1;
  b ← b + 1;
stop.
```

- a) \*  
 b) \*\*  
 c) \*\*\*  
 d) \*\*\*\*

9. Care este valoarea minimă pe care o poate avea variabila  $x$ , astfel încât instrucțiunea *while-do* următoare să nu efectueze nici o iterație?

```
intreg x, y;
x ← ?; y ← 10;
cat timp  $2 * x - 1 < y$  executa
  scrie y;
  x ← x + 1;
stop.
```

- a) 5  
 b) 0  
 c) 6  
 d) 7

10. Care dintre următoarele instrucțiuni execută un număr infinit de iterații?

a)  $x \leftarrow 100;$   
 cat timp  $x > 10$  executa  
   daca  $x \bmod 2 = 1$  atunci  
     scrie  $x \bmod 10;$   
 ■  
 $x \leftarrow x \div 10;$   
 ■

c)  
 $x \leftarrow 100; y \leftarrow 10$   
 cat timp  $x < y$  executa  
   scrie  $x;$   
    $x \leftarrow x \div 10;$   
 ■

b)

```

x ← 100; y ← 10
cat timp x>y executa
    scrie x;
    x ← x * 10 div 10 ;

```

d)

```

x ← 100;
cat timp x>10 executa
    daca x mod 2 = 1 atunci
        scrie x ;

```

```

intreg a, s;
s ← 0; a ← 0;
repete
    a ← a + 1;
    s ← s + a;
    pana cand s >= 10;
scrie s, a;
stop.

```

- a) 9 3  
b) 11 3  
c) 10 4  
d) 11 4

11. Care dintre următoarele instrucțiuni conduce la afișarea pe ecran numai a unor valori pare?

a)

```

citeste x;
repete
    daca x mod 2 = 0 atunci
        scrie x mod 10;
    x ← x div 10 ;
    pana cand x=0;

```

c)

```

citeste x;
repete
    daca x div 2 = 0 atunci
        scrie x mod 10;
    x ← x div 10 ;
    pana cand x=0;

```

```

intreg x,y;
x ← ?; y ← 10;
repete
    scrie y;
    y ← y - 1;
    pana cand x > 2*y;
stop.

```

- a) 18.  
b) 20  
c) 19  
d) 20

b)

```

citeste x; y ← 0;
repete
    daca x mod 2 = 0 atunci
        y ← y*10 + x mod 10;
    x ← x div 10 ; scrie y ;
    pana cand x=0;

```

d)

```

y ← 1;
repete
    scrie y ;
    y ← y + 2;
    pana cand y>=10;

```

14. Care este valoarea minimă pe care o poate avea variabila x, astfel încât instrucțiunea cu test final următoare să efectueze o singură iterație?

a)

```

x ← 31;
repete
    daca x mod 10 > 0 atunci
        x ← x + 2
    pana cand x mod 10 = 0

```

c)

```

x ← 100;
repete
    daca x mod 10 < 5 atunci
        x ← x + 1
    x ← x div 10 ;
    pana cand x mod 10 = 1

```

b)

```

x ← 100;
repete
    daca x mod 10 < 5 atunci
        x ← x + 1
    x ← x div 10 ;
    pana cand x > 9

```

d)

```

x ← 32;
repete
    daca x mod 10 > 0 atunci
        x ← x + 2
    pana cand x mod 10 = 1

```

12. Care dintre următoarele secvențe de instrucțiuni permite afișarea primei cifre a unui număr?

a)

```

citeste x //x>10
repete
    x ← x div 10 ;
    pana cand x>10;
scrie x;

```

c)

```

citeste x //x>10
repete
    daca x<100 atunci
        x ← x div 10
    altfel
        x ← x div 100
    pana cand x<10;
scrie x;

```

b)

```

citeste x //x>10
repete
    x ← x div 10 ;
    pana cand x<10;
scrie x;

```

d)

```

citeste x //x>10
repete
    x ← x div 10
    pana cand x>=10;
scrie x

```

15. Care dintre următoarele instrucțiuni execută un număr infinit de iterații?

16. Se consideră următoarea secvență repetitivă:

```

i:=0; j:= ?
while (i+j<=10) do begin
    i:=i+1;
    j:=j-2;
end;

```

```

i:=0;
while (i+j<=10) {
    i:=i+1;
    j:=j-2;
}

```

Valoarea minimă posibilă pentru variabila j astfel încât instrucțiunea repetitivă de mai sus să nu se execute la infinit(necontrolat) este:

- a) 1      b) 5      c) 6      d) 17      e) 10      f) 2

13. Se consideră următorul algoritm descris în pseudocod. Ce se va afișa în urma executării instrucțiunilor:

17. Dacă  $a, b$  sunt variabile de tip *integer* (varianta Pascal) / *int* (varianta C/C++), stabiliți care dintre următoarele structuri repetitive sunt ciclări infinite:

a)  $a:=10; b:=0;$   
**repeat**  
 dec(a); inc(b);  
**until**  $a=b;$

b)  $a:=1; b:=5;$   
**while**  $(a+b \leq 10)$  **do begin**  
 $a:=a+2; b:=b-2;$   
**end;**

c)  $a:=0; b:=10;$   
**while**  $(a \leq 10) \text{ and } (b \geq 0)$  **do**  
**begin**  
 inc(b); inc(a);  
**end;**

d)  $a:=0; b:=10;$   
**while**  $(a \leq 10) \text{ or } (b \geq 0)$  **do**  
 inc(b);

a)  $a:=10; b:=0;$   
**do**  
 $a--; b++;$   
**while**  $(a \neq b);$

b)  $a:=1; b:=5;$   
**while**  $(a+b \leq 10)$  {  
 $a=a+2;$   
 $b=b-2;$ }

c)  $a:=0; b:=10;$   
**while**  $((a \leq 10) \text{ \&\& } (b \geq 0))$  {  
 $b++;$   
 $a++;$ }

d)  $a:=0; b:=10;$   
**while**  $((a \leq 10) \text{ || } (b \geq 0))$  {  
 $b++;$ }

18. Se consideră următoarea secvență de program în care  $i, k, n$  sunt variabile de tip *integer* / *int* (varianta Pascal/C++), iar  $a, b, x$  sunt variabile de tip *real* / *float* (varianta Pascal/C++):

read(a,b,n);  
 $k:=0;$   
**for**  $i:=1$  **to**  $n$  **do**  
**begin**  
 read(x);  
 if ..... **then**  $k:=k+1;$   
**end;**  
 write(k);

cin>>a>>b>>n;  
 $k:=0;$   
**for**  $(i:=1; i \leq n; i++)$   
 {  
 cin>>x;  
 if (.....)  $k++;$   
 }  
 cout<<k;

Indicați cu care dintre expresiile următoare pot fi înlocuite punctele din secvența de mai sus astfel încât să se afișeze câte dintre cele  $n$  valori citite în  $x$  se găsesc în intervalul  $[a, b]$  (se presupune că  $a \leq b$ ).

a)  $(a \leq x) \text{ and } (b \geq x)$   
 b)  $(x \geq a) \text{ or } (x \leq b)$   
 c)  $\text{not}((x < a) \text{ or } (x > b))$   
 d)  $(x \leq a) \text{ and } (x \geq b)$

a)  $(a \leq x) \text{ \&\& } (b \geq x)$   
 b)  $(x \geq a) \text{ || } (x \leq b)$   
 c)  $\text{!}((x < a) \text{ || } (x > b))$   
 d)  $(x \leq a) \text{ \&\& } (x \geq b)$

19. Care dintre următoarele secvențe de program afișează pe ecran următorul șir de numere 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 (variabilele  $i$  și  $j$  sunt de tip întreg)?

a) **for**  $i:=1$  **to** 5 **do**  
**for**  $j:=1$  **to** 5 **do**  
 write(i, ' ');

a) **for**  $(i:=1; i \leq 5; i++)$   
**for**  $(j:=1; j \leq 5; j++)$   
 cout<<i<<" ";

b) **for**  $i:=1$  **to** 5 **do**  
**for**  $j:=1$  **to**  $i$  **do**  
 write(i, ' ');

c) **for**  $i:=1$  **to** 5 **do**  
**for**  $j:=1$  **to**  $i$  **do**  
 write(j, ' ');

d) **for**  $i:=1$  **to** 4 **do**  
**for**  $j:=i+1$  **to** 5 **do**  
 write(i, ' ');

b) **for**  $(i:=1; i \leq 5; i++)$   
**for**  $(j:=1; j \leq i; j++)$   
 cout<<i<<" ";

c) **for**  $(i:=1; i \leq 5; i++)$   
**for**  $(j:=1; j \leq i; j++)$   
 cout<<j<<" ";

d) **for**  $(i:=1; i \leq 4; i++)$   
**for**  $(j:=i+1; j \leq 5; j++)$   
 cout<<i<<" ";

20. Ce se va afișa în urma executării următorului program ?

var  $n, i$  : integer ;  
**begin**  
 $n:=7;$   
**for**  $i:=2$  **to**  $n-2$  **do**  
**if**  $i \bmod 2=0$  **then**  
 write(i, ' ',  $n-i$ , ' ');  
**end.**

#include<iostream.h>  
 int  $n=7, i;$   
 void main() {  
**for**  $(i:=2; i \leq n-2; i++)$   
**if**  $(i \% 2 == 0)$  cout<< i << ' ' <<  
 $n-i$  << ' ' ;  
 }

a) 2 4 6      b) 2 4 4 6      c) 2 5 4 3      d) 2 5      e) 2 5 3 4

21. Câte caractere se vor afișa în urma execuției următoarei secvențe de instrucțiuni?

var  $n, i, j$  : integer ;  
**begin**  
 $n:=5;$   
**for**  $i:=1$  **to**  $n$  **do begin**  
 write(i);  
**for**  $j:=1$  **to** 3 **do** write(j);  
**end;**  
**end.**

#include<iostream.h>  
 int  $n, i, j;$   
 void main() {  
 $n:=5;$   
**for**  $(i:=1; i \leq n; i++)$  {  
 cout<<i;  
**for**  $(j:=1; j \leq 3; j++)$  cout<<j;  
 }  
 }

a) 5      b) 6      c) 16      d) 20      e) 24

22. Un număr reprezintă o putere a lui 2 ( $2^K$ ) dacă orice divizor al său (mai mare ca 1) este par. Care dintre următoarele secvențe de instrucțiuni verifică în mod corect dacă un număr  $n$  este o putere a lui 2 ?

a)  
 $ok:=false;$   
**for**  $i:=2$  **to**  $n$  **do**  
**if**  $(n \bmod i=0) \text{ and } (i \bmod 2=0)$   
**then**  $ok:=true;$   
**if**  $ok$  **then** write('DA')  
**else** write('NU');

a)  
 $ok:=0;$   
**for**  $(i:=2; i \leq n; i++)$   
**if**  $(n \% i == 0 \text{ \&\& } i \% 2 == 0)$   $ok:=1;$   
**if**  $(ok)$  cout << "DA";  
**else** cout << "NU";

b)

```
ok:=true;
for i :=2 to n do
  if (n mod i=0)and(i mod 2=0)
    then ok:=false;
  if ok then write('DA')
    else write('NU');
```

c)

```
ok:=true;
for i :=2 to n do
  if (n mod i=0)and(i mod 2<>0)
    then ok:=false;
  if ok then write('DA')
    else write('NU');
```

d)

```
ok:=false;
for i :=2 to n do
  if (n mod i=0)and(i mod 2<>0)
    then ok:=true;
  if not ok then write('DA')
    else write('NU');
```

23. Un număr este perfect dacă suma divizorilor săi (strict mai mici decât el) este egală cu numărul respectiv. De exemplu 6 este număr perfect deoarece  $6=1+2+3$ . Care dintre următoarele secvențe de instrucțiuni verifică în mod corect dacă un număr este perfect?

a)

```
s:=0;
for i :=1 to n div 2 do
  if n mod i=0 then s:=s+i;
  if s=n then write('DA')
    else write('NU');
```

b)

```
s:=1;
for i :=2 to n div 2 do
  if n mod i=0 then s:=s+i;
  if s=n then write('DA')
    else write('NU');
```

c)

```
s:=0;
for i :=1 to n div 2 do
  if n div i=0 then s:=s+i;
  if s=n then write('DA')
    else write('NU');
```

d)

b)

```
ok:=1;
for (i:=2; i<=n; i++)
  if (n%i==0 && i%2==0) ok=0;
  if (ok) cout << "DA";
  else cout << "NU";
```

c)

```
ok:=1;
for (i:=2; i<=n; i++)
  if (n%i==0 && i%2>0) ok=0;
  if (ok) cout << "DA";
  else cout << "NU";
```

d)

```
ok:=0;
for (i:=2; i<=n; i++)
  if (n%i==0 && i%2>0) ok=1;
  if (!ok) cout << "DA";
  else cout << "NU";
```

a)

```
s:=0;
for (i:=1; i<=n/2; i++)
  if (n%i==0) s+=i;
  if (s==n) cout << "DA";
  else cout << "NU";
```

b)

```
s:=1;
for (i:=2; i<=n/2; i++)
  if (n%i==0) s+=i;
  if (s==n) cout << "DA";
  else cout << "NU";
```

c)

```
s:=0;
for (i:=1; i<=n/2; i++)
  if (n/i==0) s+=i;
  if (s==n) cout << "DA";
  else cout << "NU";
```

d)

```
s:=1;
for i :=1 to n div 2 do
  if n mod i=0 then s:=s+i;
  if s=n then write('DA')
    else write('NU');
```

```
s:=1;
for (i:=1; i<=n/2; i++)
  if (n%i==0) s+=i;
  if (s==n) cout << "DA";
  else cout << "NU";
```

24. Care dintre următoarele instrucțiuni permit afișarea unui număr de 15 caractere pe ecran ?

a)

```
for i :=1 to 3 do
  for j:=1 to 5 do write('DA')
```

b)

```
for i :=1 to 3 do begin
  write('A');
  for j:=1 to 4 do
    write('B');
end;
```

c)

```
for i :=2 to 4 do
  for j:=1 to 5 do write('A')
```

d)

```
for i :=1 to 30 do
  if i mod 2=0 then
    write('A')
```

a)

```
for (i:=1; i<=3; i++)
  for (j:=1; j <=5; j++)
    cout << "DA";
```

b)

```
for (i:=1; i<=3; i++) {
  cout << "A";
  for (j:=1; j<=4; j++)
    cout << "B";
}
```

c)

```
for (i:=2; i<=4; i++)
  for (j:=1; j<=5; j++)
    cout << "A";
```

d)

```
for (i:=1; i<=30; i++)
  if (i%2==0) cout << "A";
```

25. Ce se va afișa pe ecran în urma executării următoarei secvențe de instrucțiuni, știind că pentru variabila  $x$  au fost citite valorile 23, 25, 345, 892 și 3456 ?

```
for i :=1 to 5 do begin
  readln(x);
  if i mod 2=0 then
    write(x mod 10)
  else write( x div 10 mod 10);
end;
```

```
for (i:=1; i<=5; i++) {
  cin >> x;
  if (i%2==0) cout << x%10;
  else cout << (x/10)%10;
}
```

a) 3 2 5 9 5   b) 3 5 5 9 2   c) 2 5 5 9 5   d) 2 5 5 2 5   e) 2 5 4 2 5

26. Care dintre următoarele instrucțiuni *FOR* sunt incorecte sintactic?

a)

```
for i := 1 to n div 2 do
  write ('i');
```

b)

```
for i:= n down to n div 2 do
  write(i);
```

c)

```
for i :=1 to n / 2 do
  write('i');
```

a)

```
for (i:=1; i<=n/2; i++)
  cout << 'i';
```

b)

```
for (i:=1 to n) do
  cout << i;
```

c)

```
for (i:=1; i <= n div 2; i++)
  cout << 'i';
```

d)  

```
for i = 1 to n div 2 do
  write(i);
```

27. Care dintre următoarele instrucțiuni *FOR* calculează în mod corect suma numerelor pare mai mici sau egale cu numărul  $n$ ?

a)  

```
s:=0;
for i := 1 to n do
  if i mod 2=0 then s:=s+i;
```

b)  

```
s:=0;
for i:= n downto 2 do
  if i mod 2=0 then s:= i;
```

c)  

```
s:=0;
for i:= 1 to n div 2 do
  s:=s+2*i;
```

d)  

```
s:=0;
for i :=(n-1) downto 2 do
  if i mod 2=0 then s:=s+i;
```

28. Care dintre următoarele instrucțiuni *FOR* permit afișarea corectă a tuturor numerelor impare mai mici sau egale cu numărul  $n$ ?

a)  

```
for i := 1 to n do begin
  write(i); i:=i+2;
end;
```

b)  

```
for i := n downto 1 do
  if i mod 2=1 then write(i);
```

c)  

```
for i = 0 to (n-1) div 2 do
  write(i*2+1);
```

d)  

```
for i = 1 to n div 2 do
  write(i*2+1);
```

d)  

```
for (i:=1; i<=n/2; i++)
  cout << i;
```

a)  

```
s:=0;
for (i=1; i<=n; i++)
  if (i%2==0) s+=i;
```

b)  

```
s:=0;
for (i=n; i>=2; i--)
  if (i%2==0) s=i;
```

c)  

```
s:=0;
for (i=1; i<=n/2;i++)
  s+=2*i;
```

d)  

```
s:=0;
for(i=n-1; i>=2; i--)
  if (i%2==0) s+=i;
```

a)  

```
for (i=1; i<=n; i+=3)
  cout << i;
```

b)  

```
for (i=n; i>=1; i--)
  if (i%2==1) cout << i;
```

c)  

```
for (i=0; i<=(n-1)/2;i++)
  cout << 2*i+1;
```

d)  

```
for (i=1; i<=n/2; i++)
  cout << 2*i+1;
```

29. Care dintre următoarele instrucțiuni *FOR* permit afișarea unor caractere dispuse pe mai multe linii sub forma unui triunghi?

a)  

```
for i := 1 to 5 do begin
  writeln;
  for j:=1 to i write(i);
end;
```

b)  

```
for i := 1 to 5 do
  for j:=1 to i begin
    write(i);
    writeln;
  end;
```

c)  

```
for i := 1 to 5 do begin
  writeln;
  for j:=1 to i
    writeln(i);
end;
```

d)  

```
for i := 1 to 5 do begin
  write;
  for j:=1 to i begin
    writeln(i);
  end;
```

a)  

```
for (i=1; i<=5; i++) {
  cout << endl;
  for (j=1; j<=i; j++) cout <<i;
}
```

b)  

```
for (i=1; i<=5; i++)
  for (j=1; j<=i; j++) {
    cout << i;
    cout << endl;
  }
```

c)  

```
for (i=1; i<=5; i++) {
  cout << endl;
  for (j=1; j<=i; j++)
    cout << i << endl;
}
```

d)  

```
for (i=1; i<=5; i++) {
  for (j=1; j<=i; j++)
    cout << i << endl;
}
```

30. Ce se va afișa la executarea următoarelor instrucțiuni?

```
k:=1;
while k<=7 do begin
  write(k, ' ');
  if k mod 2 = 1 then k:=k+2
  else k:=k+1;
end;
```

a) 1 2 3 4 5 6 7    b) 1 3 5 7    c) 1 3 4 6 7    d) 1 2 4 5 6

```
k:=1;
while (k<=7) {
  cout << k << ' ';
  if (k%2==1) k+=2; else k++;
}
```

31. Ce se va afișa la executarea următoarelor instrucțiuni?

```
a:=1;
while a mod 5 <>0 do begin
  write(a mod 5);
  a:=a+1;
end;
```

a) 0 1 2 3 4    b) 1 2 3 4 0

```
a:=1;
while (a%5!=0) {
  cout << a%5;
  a++;
}
```

c) 1 2 3 4    d) 2 3 4 0

32. Ce se va afișa la executarea următoarelor instrucțiuni?

<pre>x:=3.00; y:=2.00; while x*y&lt;100 do x:=x*y; write(x :2 :0, ' ', y :2 :0);</pre>	<pre>x=3.00; y=2.00; while (x*y&lt;100) x*=y; printf("%2.0f %2.0f", x, y);</pre>
--	--

- a) 48 2                      b) 48.0 2.0                      c) 96.0 2.0                      d) 96 2

33. Ce se va afișa la executarea următoarelor instrucțiuni?

<pre>x:=32034; while x mod 10&lt;&gt;0 do     x:=x div 10; write(x, ' ', x div 10 mod 10);</pre>	<pre>x=32034; while (x%10&gt;0) x/=10; cout &lt;&lt; x &lt;&lt; ' ' &lt;&lt; (x/10)%10;</pre>
--	---

- a) 3 0                      b) 320 2                      c) 320 0                      d) 3 3

34. Identificați care dintre următoarele instrucțiuni execută un număr infinit de iterații.

a)  

```
x:=1; k:=2*x;
while k>0 do begin
    write(k+x);
    k:=k div 1;
end;
```

b)  

```
a:=12;
while a mod 10 <> 0 do begin
    write(a);
    a:=a div 10;
end;
```

c)  

```
x:=231;
while x<>5 do begin
    writeln(x);
    x:=x * (x mod 10);
end;
```

d)  

```
a:=1;
while a mod 10 = 5 do begin
    writeln(a);
    write(a div 10);
end;
```

a)  

```
x:=1; k:=2*x;
while (k>0) {
    cout << k+x;
    k/=1;
}
```

b)  

```
a:=12;
while (a%10>0) {
    cout << a;
    a/=10;
}
```

c)  

```
x:=231;
while (x!=5) {
    cout << x;
    x*=x%10;
}
```

d)  

```
a:=1;
while (a%10==5) {
    cout << a;
    cout << a/10;
}
```

35. Ce se va afișa la executarea următoarelor instrucțiuni?

<pre>a:=0; b:=10; repeat     a:=a+2; b:=b-2;     write(a, ' ', b, ' '); until a&gt;b;</pre>	<pre>a:=0; b:=10; do {     a+=2; b-=2;     cout &lt;&lt; a &lt;&lt; ' ' &lt;&lt; b &lt;&lt; ' '; } while (a&lt;=b);</pre>
---	---

- a) 2 8 4 6 6 4                      b) 0 10 2 8 4 6                      c) 2 8 4 6                      d) 0 10 2 8 4 6 6 4

36. Ce se va afișa la executarea următoarelor instrucțiuni?

<pre>p:=1; repeat     p:=p*2;     write(p); until p mod 10=6;</pre>	<pre>p:=1; do {     p*=2;     cout &lt;&lt; p; } while (p%10!=6);</pre>
---	---

- a) 2 4 8 6                      b) 2486                      c) 2 4 8 16                      d) 24816

37. Ce se va afișa la executarea următoarelor instrucțiuni?

<pre>j:=1; repeat     write(j, ' ');     if j mod 2 =0 then j:=j+1     else j:=j*2; until j&gt;10;</pre>	<pre>j:=1; do {     cout &lt;&lt; j &lt;&lt; ' ';     if (j%2==0) j++;     else j*=2; } while (j&lt;=10);</pre>
--	---

- a) 1 2 3 6                      b) 1 3 6                      c) 1 2 4 6                      d) 1 2 3 6 7

38. Ce se va afișa la executarea următoarelor instrucțiuni?

<pre>a:=2; repeat     write(a:2);     a:=sqr(a); until a&gt;20;</pre>	<pre>a:=2; do {     printf("%2d", a);     a*=a; } while (a&lt;=20);</pre>
---	---

- a) 2 4 816                      b) 2 416                      c) 2 4 8 16                      d) 24816

39. Indicați care dintre următoarele instrucțiuni execută un număr infinit de iterații.

<pre>a) j:=1234; i:=10; repeat     write(j);     j:= i div 10; until j&gt;10;</pre>	<pre>a) j:=1234; i:=10; do {     cout &lt;&lt; j;     j=i/10; } while (j&lt;=10);</pre>
---	---

b)

```

a:=2;
repeat
  a:=a*2;
  writeln(a);
until a>20;

c)

a:=2;
repeat
  writeln(a);
  if a < 1000 then a:=a*2;
until a<1;

d)

a:=1;
repeat
  write(a);
  a:=trunc(sqrt(a));
until a>0;

```

b)

```

a:=2;
do {
  a*=2;
  cout << a << endl;
} while (a<=20);

c)

a:=2;
do {
  cout << a << endl;
  if (a<1000) a*=2;
} while (a>=1);

d)

a:=1;
do {
  cout << a;
  a=(int)sqrt((double)a);
} while (a<=0);

```

### 1.2.2 Teste cu itemi semiobiectivi

1. Se consideră următorul algoritm reprezentat în pseudocod:

```

1  întreg a, b, c, i ;
2  citește a, b ;
3  c ← 0;
4  pentru i ← a, b executa
5  |   dacă i mod 2=0 atunci
6  |   |   c ← c + 1;
7  |   |
8  |   |
9  |   dacă c > 0 atunci scrie c
10  |   altfel scrie "Nu exista"
11  |   stop.

```

- a) Ce se va afișa pentru valorile  $a=3$  și  $b=10$ ? Dar pentru  $a=4$  și  $b=12$ ?
- b) Dați un exemplu de valori pentru datele de intrare astfel încât algoritmul să afișeze mesajul «Nu exista»
- c) Creați un enunț de problemă a cărei rezolvare este algoritmul prezentat.
- d) Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.

2. Se consideră următorul algoritm reprezentat în pseudocod:

```

1  întreg n, x, nr, i ;
2  citește n ;
3  nr ← 0;

```

- a) Ce se va afișa pentru  $n=4$  și șirul de valori 3, 5, 6, 10?

```

4  pentru i ← 1, n executa
5  |   citește x;
6  |   dacă (x<n) or (x>2*n) atunci
7  |   |   nr ← nr + x;
8  |   |
9  |   |
10  |   scrie nr;
11  |   stop.

```

- b) Dați un exemplu de set de date de intrare pentru care se va afișa valoarea 0.
- c) Dați un exemplu de set de date de intrare pentru care se va afișa suma celor  $n$  numere citite.
- d) Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.

3. Se consideră următorul algoritm reprezentat în pseudocod:

```

1  întreg x, s, i, j ;
2  pentru i ← 1, 3 executa
3  |   citește x; s ← 0;
4  |   pentru j ← 1, 2 executa
5  |   |   s ← s + x;
6  |   |
7  |   scrie s;
8  |   stop.
9  stop.

```

- a) Ce se va afișa pentru șirul de valori 1, 2, 3?
- b) Dați un exemplu de set de date de intrare pentru care se vor afișa trei valori multipli de 3.
- c) De câte ori se efectuează operația de atribuire  $s \leftarrow s + x$ ?
- d) Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.

4. Se consideră următorul algoritm reprezentat în pseudocod:

```

1  întreg i, j, s, x ;
2  pentru i ← 1, 5 executa
3  |   citește x; s ← 0;
4  |   pentru j ← 1, i executa
5  |   |   s ← s + x;
6  |   |
7  |   scrie s;
8  |   stop.
9  stop.

```

- a) Ce se va afișa pentru șirul de valori 6, 5, 4, 2, 1?
- b) Dați un exemplu de set de date de intrare pentru care se vor afișa cinci valori consecutive.
- c) De câte ori se efectuează operația de atribuire  $s \leftarrow s + x$ ?
- d) Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.

5. Se consideră următorul algoritm reprezentat în pseudocod:

```

1  întreg i, j, n ;
2  citește n;
3  pentru i ← 1, n executa
4  |   pentru j ← 1, n executa
5  |   |   scrie (i-1)*n + j;
6  |   |
7  |   |
8  |   stop.

```

- a) Ce se va afișa pentru  $n=3$ ?
- b) De câte ori contorul  $i$  va lua valoarea  $n$ ?
- c) De câte ori contorul  $j$  va lua valoarea  $n$ ?
- d) Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.
- e) Realizați un enunț de problemă a cărei rezolvare este chiar algoritmul alăturat

6. Se consideră următorul algoritim reprezentat în pseudocod:

```

1  | 1 | intreg a, nr;
2  | 2 | citește a;
3  | 3 | nr ← 0;
4  | 4 | cat timp a ≠ 0 executa
5  | 5 |   |  |  |  |  |
6  | 6 |   |  |  |  |  |  |  |
7  | 7 |   |  |  |  |  |  |  |
8  | 8 |   |  |  |  |  |  |  |
9  | 9 |   |  |  |  |  |  |  |
10 | 10 | scrie nr; stop.

```

- Ce se va afișa la introducerea șirului de valori 2 4 5 7 4 0?
- Dați un exemplu de valori pentru datele de intrare astfel încât algoritmul să afișeze valoarea 0.
- Creați un enunț de problemă a cărei rezolvare este algoritmul prezentat.
- Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.

7. Se consideră următorul algoritim reprezentat în pseudocod:

```

1  | 1 | intreg x, nr;
2  | 2 | citește x;
3  | 3 | nr ← 0;
4  | 4 | cat timp x ≠ 0 executa
5  | 5 |   |  |  |  |  |
6  | 6 |   |  |  |  |  |  |
7  | 7 |   |  |  |  |  |  |
8  | 8 |   |  |  |  |  |  |
9  | 9 |   |  |  |  |  |  |
10 | 10 | scrie nr; stop.

```

- Ce se va afișa dacă  $x=7$ ? Dar pentru  $x=13$ ?
- Dați un exemplu de valoare  $x$  astfel încât să se afișeze valoarea 1.
- Creați un enunț de problemă a cărei rezolvare este algoritmul prezentat.
- Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.

8. Se consideră următorul algoritim reprezentat în pseudocod:

```

1  | 1 | intreg x, nr, y;
2  | 2 | citește x; nr ← 0;
3  | 3 | cat timp x ≠ 0 executa
4  | 4 |   |  |  |  |  |
5  | 5 |   |  |  |  |  |  |
6  | 6 |   |  |  |  |  |  |
7  | 7 |   |  |  |  |  |  |
8  | 8 |   |  |  |  |  |  |
9  | 9 |   |  |  |  |  |  |
10 | 10 | scrie nr; stop.

```

- Ce se va afișa pentru șirul de valori 2 4 6 5 7 4 3 0?
- Dați un exemplu de șir de valori pentru care se afișează valoarea 0.
- Dați un exemplu de șir de valori pentru care valoarea afișată va fi egală cu numărul de numere introdus.
- Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.

9. Se consideră următorul algoritim reprezentat în pseudocod:

```

1  | 1 | intreg x, nr;
2  | 2 | citește x; nr ← 0
3  | 3 | repeta
4  | 4 |   |  |  |  |  |
5  | 5 |   |  |  |  |  |  |

```

- Ce se va afișa pentru  $x=2$ ?
- Ce valori poate lua la intrare  $x$  pentru ca la final să se afișeze 10?
- Dați un exemplu de valoare pentru

```

6  | 6 |   |  |  |  |  |  |
7  | 7 |   |  |  |  |  |  |
8  | 8 |   |  |  |  |  |  |
9  | 9 |   |  |  |  |  |  |
10 | 10 | scrie nr; stop.

```

- $x$  astfel încât la final să nu fie afișată o valoare divizibilă cu 10.
- Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.

10. Se consideră următorul algoritim reprezentat în pseudocod:

```

1  | 1 | intreg x, nr;
2  | 2 | citește x; nr ← 0;
3  | 3 | repeta
4  | 4 |   |  |  |  |  |
5  | 5 |   |  |  |  |  |  |
6  | 6 |   |  |  |  |  |  |
7  | 7 |   |  |  |  |  |  |
8  | 8 |   |  |  |  |  |  |
9  | 9 |   |  |  |  |  |  |
10 | 10 | scrie nr;
11 | 11 | stop.

```

- Ce se va afișa pentru șirul de valori 4 6 5 11 15?
- Dați un exemplu de șir de valori pentru care se afișează valoarea 0.
- Ce valoare trebuie să aibă la intrare  $x$  pentru ca structura *repeta* să efectueze 3 iterații?
- Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.

11. Se consideră următorul algoritim:

```

1  | 1 | intreg x, p, i;
2  | 2 | i ← 0; p ← 1;
3  | 3 | citește x;
4  | 4 | cat timp (x≠0) executa
5  | 5 |   |  |  |  |  |
6  | 6 |   |  |  |  |  |  |
7  | 7 |   |  |  |  |  |  |
8  | 8 |   |  |  |  |  |  |
9  | 9 |   |  |  |  |  |  |
10 | 10 | scrie i;
11 | 11 | stop.

```

- Ce valori vor fi afișate dacă se vor introduce, în ordine, numerele: 2, 3, 4, 5, 0;
- Determinați un set de date de intrare pentru care ultimele două numere afișate să fie egale.
- Modificați algoritmul, fără să introduceți noi variabile sau instrucțiuni, astfel încât să se afișeze succesiv media aritmetică a numerelor citite.

12. Se consideră următorul algoritim:

```

1  | 1 | intreg n, i, nr, x, s;
2  | 2 | citește n; s ← 0;
3  | 3 | pentru i ← 1, n executa
4  | 4 |   |  |  |  |  |
5  | 5 |   |  |  |  |  |  |
6  | 6 |   |  |  |  |  |  |
7  | 7 |   |  |  |  |  |  |
8  | 8 |   |  |  |  |  |  |
9  | 9 |   |  |  |  |  |  |
10 | 10 | scrie s

```

- Ce valoare va fi afișată pentru  $n=5$  și numerele: 222, 2043, 29, 2, 20035.
- Determinați un set de date de intrare pentru care valoarea afișată va fi egală cu suma numerelor citite?
- Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.
- Rescrieți instrucțiunile repetitive ale algoritmului folosind numai instrucțiunea condiționată posterior.



13. Se consideră următorul algoritm:

```

1  integ n,i,j;
2  citește n;
3
4  pentru i ← 1, n executa
5      pentru j ← 1, n-i executa
6          scrie ' ';
7
8      pentru j ← n-i+1, n executa
9          scrie '*';
10
11     scrie salt la linie noua
12

```

- Ce se va afișa pentru valoarea  $n=4$ ?
- Scrieți un nou algoritm care va afișa pe ecran imaginea simetrică ("răsturnată"), față de axa orizontală, a celei descrise de algoritm.
- Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.
- Scrieți un nou algoritm care va afișa imaginea simetrică față de axa verticală.

14. Se consideră următorul algoritm:

```

1  integ n,x,nr,j; logic ok;
2  citește n;
3  nr ← 0; x ← n; ok ← True;
4
5  repeta
6      nr ← nr*10 + n mod 10
7      n ← [n / 10];
8      pana cand n=0;
9
10     pentru j ← 2, [√nr] executa
11         dacă nr%j=0 atunci ok=False
12
13     dacă ok atunci
14         scrie nr/x {cu 2 zecimale}
15     altfel
16         scrie x/nr {cu 2 zecimale}

```

- Ce valoare va fi afișată pentru  $n=31$ ?
- Determinați o valoare a lui  $n$  pentru care se obține o egalitate între valoarea datei de ieșire și cea de intrare.
- Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.
- Rescrieți instrucțiunile repetitive ale algoritmului folosind numai instrucțiunea condiționată anterior.
- Determinați două valori ale datei de intrare cu proprietatea: ambele vor conduce la afișarea valorii 1.00, dar din considerente diferite.

15. Se consideră următorul algoritm:

```

1  integ n,i,j; logic ok;
2  citește n;
3  pentru i ← 2, n-2 executa
4      ok ← True;
5
6      pentru j ← 2, [√i] executa
7          dacă i % j=0 atunci ok=False
8
9
10     pentru j ← 2, [√n-i] executa
11         dacă (n-i) mod j = 0 atunci
12             ok ← False;
13

```

- Ce valori vor fi afișate pentru  $n=10$ ?
- Determinați o valoare pentru care nu se va face nici o afișare.
- Determinați o valoare pentru care vor fi afișate trei perechi de numere.
- Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.
- Modificați algoritmul fără a introduce noi instrucțiuni sau

```

14  dacă ok atunci scrie i, n-i;
15
16

```

- variabile astfel încât o pereche de numere să nu fie afișată de două ori.
- Introduceți o instrucțiune alternativă cu scopul de a elimina operațiile inutile ale algoritmului.

16. Se consideră următorul algoritm:

```

1  integ n, i, max1, max2, x, c;
2  citește n;
3  max1 ← -1; max2 ← -1;
4  pentru i ← 1, n executa
5      citește x; c ← x mod 10;
6      dacă c > max1 atunci
7          max2 ← max1; max1 ← c;
8      altfel
9          dacă (c>max2) and (c≠max1)
10             atunci max2 ← c;
11
12
13
14  scrie max2, max1;
15  stop.

```

- Ce valori vor fi afișate pentru  $n=5$  și numerele: 322, 3043, 39, 9, 30035?
- Pentru  $n=4$ , determinați un set de date de intrare astfel încât una din valorile afișate să fie -1?
- Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.
- Rescrieți algoritmul astfel încât să fie afișat numărul de apariții al celei mai mari cifre a unităților numerelor introduse.

17. Se consideră următorul algoritm:

```

1  integ n, i, c, x;
2  citește n;
3  pentru i ← 1, n executa
4      citește x;
5      c ← 0;
6
7      repeta
8          c ← c*10 + x mod 10;
9          x ← [x / 100];
10         pana cand x = 0;
11         scrie c;
12
13  stop.

```

- Ce valori vor fi afișate pentru  $n=5$  și numerele: 542, 32103, 3, 91, 21035?
- Pentru  $n=4$ , determinați un set de date de intrare format din valori distincte pentru care toate valorile afișate să fie egale?
- Realizați programul Pascal/C/C++ corespunzător algoritmului prezentat.
- Rescrieți algoritmul astfel încât să conțină doar instrucțiuni repetitive condiționate anterior.

### 1.2.3 Probleme rezolvate

1. Să se determine cel mai mare număr care se poate forma cu ajutorul cifrelor unui număr natural citit de la tastatură.

Exemplu: pentru  $nr=30027$  se va afișa 73200

**Soluție:** Algoritmul realizează numărarea aparițiilor fiecărei cifre în ordine descrescătoare de la 9 la 0 și afișarea succesivă a acestora.

```

1 var nr,ap,x,i,c:longint;
2 begin
3   readln(nr);
4   for c:=9 downto 0 do begin
5     x:=nr;
6     ap:=0;
7     repeat
8       if x mod 10 = c then
9         inc(ap);
10      x:=x div 10;
11    until x=0;
12    for i:=1 to ap do write(c);
13  end;
14 end.
15
#include <iostream.h>
long nr,x,i,ap,c;
void main(){
  cin >> nr;
  for (c=9; c>=0; c--) {
    x:=nr;
    ap:=0;
    do {
      if (x%10==c) ap++;
      x=x/10;
    } while (x>0);
    for (i=1; i<=ap; i++)
      cout << c;
  }
}

```

2. Să se determine cel mai mic număr care se poate forma cu cifrele unui număr natural citit de la tastatură. *Exemplu:* pentru  $nr=30027$  se va afișa 20037

**Soluție:** Algoritmul numără aparițiile fiecărei cifre de la 0 la 9 efectuându-se tipărirea succesiv. Cifrele egale cu 0 vor fi afișate numai după o cifră nenulă.

```

1 var nr,ap,ap0,x,i,c,w:longint;
2 begin
3   readln(nr); w:=0;
4   for c:=0 to 9 do begin
5     x:=nr; ap:=0; ap0:=0;
6     repeat
7       if x mod 10 = c then
8         inc(ap);
9       if x mod 10 = 0 then
10        inc(ap0);
11      x:=x div 10;
12    until x=0;
13    if (c<>0) and (ap>0) then begin
14      if w=0 then begin
15        write(c);
16        for i:=1 to ap0 do
17          write(0);
18        dec(ap); inc(w);
19      end;
20      for i:=1 to ap do write(c);
21    end;
22  end;
23 end.
#include <iostream.h>
long nr,x,i,w,ap0,ap,c;
void main(){
  cin>>nr; w=0;
  for (c=0; c<=9; c++){
    x:=nr; ap:=0; ap0:=0;
    do {
      if (x%10==c) ap++;
      if (x%10==0) ap0++;
      x=x/10;
    } while (x>0);
    if (c!=0 && ap){
      if (!w) {
        cout << c;
        for (i=1; i<=ap0; i++)
          cout << 0;
        ap--; w++;
      }
      for(i=1; i<=ap; i++)
        cout << c;
    }
  }
}

```

3. Se consideră un număr real pozitiv citit de la tastatură. Să se alcătuiască un program care rotunjește partea întreagă a acestuia la cel mai apropiat întreg divizibil cu  $10^p$ , unde  $p$  este o cifră din baza 10. Se consideră că numărul de cifre ale lui  $nr$  este mai mare decât  $p$ .

*Exemplu:* pentru numărul 13087.3 și  $p=2$  se va afișa 13100.3

**Soluție:** Algoritmul prelucurează partea întreagă a numărului dat. Se calculează valoarea  $10^p$ , iar restul modulo  $10^p$  indică tipul rotunjirii: prin adaos sau trunchiere.

```

1 var nr,f:real;
2   i,ex,x,p:longint;
3 begin
4   readln(nr);
5   readln(p);
6   x:=trunc(nr);
7   f:=frac(nr);
8   ex:=1;
9   for i:=1 to p do ex:=ex*10;
10  if x mod ex>5*(ex div 10) then
11    x:=x+(ex-x mod ex)
12  else
13    x:=x-x mod ex;
14  nr:=x+f;
15  writeln(nr:0:2);
16 end.
#include <stdio.h>
#include <math.h>
float nr,f;
long i,p,x,ex;
void main(){
  scanf("%f%d",&nr,&p);
  x=(long)nr;
  f=nr-(float)x;
  p=pow(10,p);
  if (x%p>5*(p/10)) x+=p-x%p;
  else x-=x%p;
  nr=(float)x+f;
  printf("%.2f\n",nr);
}

```

4. Realizați un program care primește un număr real pozitiv și îi trunchiază partea fracționară astfel încât cifrele rămase formează un șir monoton.

*Exemplu:* numărul 3.563289 devine 3.56.

**Soluție:** Algoritmul realizează o înmulțire succesivă a numărului cu 10 cât timp cifra zecilor și cifra unităților curente mențin monotonia inițială. Variabila  $ok1$  codifică monotonia inițială, iar  $ok2$  monotonia curentă. În final, numărul este împărțit la 10 de tot atâtea ori de câte ori a fost înmulțit.

```

1 var x:real;
2   i,k,y:longint;
3   ok1,ok2:boolean;
4 begin
5   readln(x);
6   x:=x*10; i:=1;
7   y:=trunc(x*10);
8   ok1:=y mod 10>y div 10 mod
9   10;
10  ok2:=ok1;
11  while ok1=ok2 do begin
12    inc(i);
13    x:=x*10; y:=trunc(x*10);
14    ok2:=y mod 10>y div 10 mod 10;
15  end;
16  x:=y div 10;
#include <stdio.h>
#include <math.h>
float x;
long i,k,ok1,ok2,y;
void main() {
  scanf("%f",&x);
  x=x*10.0; i=1;
  y=(long) (x*10.0);
  ok1=(y%10) > (y/10%10);
  ok2=ok1;
  while (ok1==ok2) {
    i++;
    x*=10.0; y=(long) (x*10.0);
    ok2=(y%10) > (y/10%10);
  }
  x=y/10;
}

```

```

17 for k:=1 to i do x:=x/10;
18 writeln(x:0:i)
19 end.
20

```

```

for (k=1; k<=i; k++) x/=10;
printf("%.f", (int)i, x);
}

```

5. Se consideră un șir de  $n$  numere naturale. Să se afișeze numărul din șir a cărui descompunere conține un număr maxim de factori primi distincți.

*Exemplu:* pentru  $n=4$  și numerele 23 21 60 71 se va afișa 60 (conține trei factori primi distincți  $2^2 \cdot 3 \cdot 5$ )

**Soluție:** Numerele vor fi descompuse succesiv în factori primi făcându-se, după caz, actualizarea valorii cu număr maxim de factori distincți.

```

1 var
2   n,maxf,maxn,nrf:longint;
3   xx,f,i,x:longint;
4   ok:boolean;
5 begin
6   readln(n); maxf:=0;
7   for i:=1 to n do begin
8     nrf:=0; f:=2;
9     readln(x); xx:=x;
10    repeat
11      ok:=false;
12      while x mod f=0 do begin
13        x:=x div f;
14        ok:=true;
15      end;
16      inc(f);
17      if ok then inc(nrf);
18    until n<f;
19    if nrf>maxf then begin
20      maxf:=nrf;
21      maxn:=xx;
22    end;
23  end;
24  writeln(maxn);
25 end.

```

```

#include <iostream.h>
long n,maxf,maxn,nrf,xx,f,i,x;
int ok;
void main() {
  cin >> n; maxf=0;
  for(i=1; i<=n; i++) {
    nrf=0; f=2;
    cin >> x; xx=x;
    do {
      ok=0;
      while (x%f==0) {
        x=x/f;
        ok=1;
      }
      f++;
      if (ok) nrf++;
    }
    while (n>=f);
    if (nrf>maxf) {
      maxf=nrf;
      maxn=xx;
    }
  }
  cout << maxn;
}

```

6. Se consideră un șir de  $n$  numere naturale. Să se afișeze fracția subunitară ireductibilă care se poate forma din numărul minim și numărul maxim din șir.

*Exemplu:* pentru  $n=4$  și numerele 3 2 5 8 se va afișa  $1/4$ .

**Soluție:** Se determină minimul și maximul din șir. Pentru a afișa fracția sub formă ireductibilă, vom împărți numitorul și numărătorul la c.m.m.d.c.

```

1 var i,min,max,x,y,n:longint;
2 begin
3   readln(n);
4   readln(min);
5   max:=min;

```

```

#include <iostream.h>
long i,min,max,x,y,n;
void main() {
  cin >> n >> min; max=min;
  for (i=1; i<=n; i++) {

```

```

6 for i:=1 to n-1 do begin
7   readln(x);
8   if x<min then min:=x;
9   if x>max then max:=x;
10 end;
11 x:=min; y:=max;
12 while x<>y do
13   if x>y then x:=x-y
14   else y:=y-x;
15 write(min div x, '/', max div x)
16 end.

```

```

cin >> x;
if (x<min) min=x;
if (x>max) max=x;
}
x=min; y=max;
while (x!=y)
  if (x>y) x-=y;
  else y-=x;
cout<< min/y << "/" << max/y;
}

```

7. Se citește un număr natural de cel mult 8 cifre. Să se formeze un alt număr din cifrele situate pe poziții impare (de la stânga spre dreapta).

*Exemplu:* pentru numărul 1234 se va afișa 13.

**Soluție:** După citirea numărului  $n$  se va forma oglinditul acestuia notat cu  $x$  (cu cifrele de la dreapta spre stânga). În final, numărul căutat se va forma din cifrele lui  $x$  situate pe poziții impare, începând cu cifra unităților.

```

1 var n,x:longint;
2 begin
3   readln(n); x:=0;
4   repeat
5     x:=x*10+n mod 10
6     n:=n div 10;
7   until n=0;
8   repeat
9     n:=n*10+ x mod 10;
10    x:=x div 100;
11  until x=0;
12  write(n);
13  end.
14

```

```

#include <iostream.h>
long n,x;
void main() {
  cin >> n; x=0;
  do {
    x=x*10 + n%10;
    n=n/10;
  } while (n!=0);
  do {
    n=n*10 + x%10;
    x=x/100;
  } while (x!=0);
  cout << n;
}

```

8. Se consideră un număr întreg citit de la tastatură. Să se afișeze cel mai apropiat număr prim față de acesta.

*Exemplu* pentru  $n=22$  se va afișa numărul 23, iar pentru numărul 20 se va afișa 19.

**Soluție:** Se determină atât numărul prim cel mai mare mai mic decât  $n$  și cel mai mic mai mare decât  $n$  și se va afișa cel mai apropiat. Pentru aceasta s-au folosit structurile repetitive condiționate posterior al căror test verifică identificarea unui număr prim.

```

1 var i,x,n1,n2:longint;
2   ok:boolean;
3 begin
4   readln(x); n1:=x;
5   repeat
6     inc(n1);

```

```

#include <iostream.h>
void main() {
  long i,x,n1,n2,ok;
  cin >> x;
  n1=x;
  do {

```

```

7  ok:=true;
8  for i:=2 to trunc(sqrt(n1))
9  do
10   if n1 mod i=0 then
11     ok:=false;
12 until ok;
13 n2:=x+1;
14 repeat
15   dec(n2);
16   ok:=true;
17   for i:=2 to trunc(sqrt(n2)) do
18     if n2 mod i=0 then
19       ok:=false;
20 until ok;
21 if x-n2<n1-x then writeln(n2)
22   else writeln(n1);
23 end.
24
n1++; ok=1;
for (i=2; i*i<=n1; i++)
  if (n1%i==0) ok=0;
} while (!ok);
n2=x+1;
do {
  n2--; ok=1;
  for (i=2; i*i<=n2; i++)
    if (n2%i==0)
      ok=0;
} while (!ok);
if (x-n2<n1-x)
  cout << n2;
else cout << n1;
}

```

9. Scrieți în toate modurile posibile ca sumă de numere naturale consecutive, un număr natural citit de la tastatură.

**Exemplu :** Pentru  $n=9$  se va afișa : 5 + 4; 2 + 3 + 4.

**Soluție:** Se generează perechi de numere  $(i,j)$  verificându-se dacă suma numerelor dintre ele este chiar  $n$ , adică  $(j+i)*(j-i+1)/2=n$ .

```

1  var i,j,k,n:longint;
2  begin
3    readln(n);
4    for i:=1 to n div 2 do
5      for j:=i+1 to n do
6        if (j+i)*(j-i+1)/2=n then
7          begin
8            for k:=i to j-1 do
9              write(k,'+');
10             writeln(j,'=',n);
11           end;
12 end.

#include <iostream.h>
long i,j,k,n;
void main() {
  cin >> n;
  for (i=1; i<=n/2; i++)
    for (j=i+1; j<=n; j++)
      if ((j+i)*(j-i+1)/2==n) {
        for (k=i; k<j; k++)
          cout << k << "+";
        cout << j << "=" << n << endl;
      }
}

```

10. Se consideră un șir de  $n$  ( $n<10$ ) numere naturale. Să se verifice dacă numărul format din primele cifre ale acestora este un palindrom.

**Exemplu:** pentru  $n=5$  și numerele 123, 435, 92, 4002, 10 se obține numărul 14941 care este palindrom.

**Soluție:** Se determină succesiv cifrele semnificative ale numerelor din șir prin împărțiri la 10 până când se ajunge la valori mai mici sau egale cu 9. Cu aceste cifre se formează un nou număr care se verifică dacă este palindrom.

```

1  var nr,nr1,nr2,n,x,i:longint;
2  begin
3    #include <iostream.h>
4    long nr,nr1,nr2,n,x,i;

```

```

3  readln(n); nr:=0;
4  for i:=1 to n do begin
5    readln(x);
6    while x>9 do x:=x div 10;
7    nr:=nr*10+x;
8  end;
9  nr1:=nr; nr2:=0;
10 while nr1<>0 do begin
11   nr2:=nr2*10+nr1 mod 10;
12   nr1:=nr1 div 10;
13 end;
14 if nr2=nr then
15   writeln(nr,'este palindrom')
16 else
17   writeln(nr,'nu e palindrom')
18 end.
19
void main() {
  cin >> n; nr:=0;
  for (i=1; i<=n; i++) {
    cin >> x;
    while (x>9) x/=10;
    nr:=nr*10+x;
  }
  nr1=nr; nr2=0;
  while (nr1!=0) {
    nr2=nr2*10 + nr1%10;
    nr1/=10;
  }
  if (nr2==nr)
    cout<<nr<<"este palindrom";
  else
    cout<<nr<<"nu e palindrom";
}

```

11. Se consideră un număr natural de cel mult 8 cifre. Să se scrie acest număr ca sumă formată doar din termeni egali cu 3 și 5. În situația în care nu există soluție se va afișa mesajul "IMPOSIBIL".

**Exemplu:** pentru  $n=7$  se va afișa IMPOSIBIL iar pentru  $n=16$  se va afișa 5+5+3+3;

**Soluție:** algoritmul de rezolvare al problemei determină numărul termenilor egali cu 5 astfel încât restul să fie divizibil cu 3. În caz contrar problema nu admite soluție.

```

1  var x,n5,n3,i:longint;
2  begin
3    read(x); n5:=0; n3:=0;
4    while (x mod 3<>0) and (x>=5) do
5      begin
6        inc(n5);
7        x:=x-5;
8      end;
9    n3:=x div 3;
10   if x mod 3<>0 then
11     write('IMPOSIBIL')
12   else begin
13     for i:=1 to n5 do
14       write(5,'+');
15     for i:=1 to n3 do
16       write(3,'+');
17   write(#8,' ');
18 end;
19 end.

#include <iostream.h>
long x,i,n3,n5;
void main() {
  cin >> x; n5=0; n3=0;
  while (x%3 && x>=5) {
    n5++;
    x-=5;
  }
  n3=x/3;
  if (x%3)
    cout << "IMPOSIBIL";
  else {
    for (i=1; i<=n5; i++)
      cout << 5 << "+";
    for (i=1; i<=n3; i++)
      cout << 3 << "+";
  }
}

```

12. Realizați un program care afișează pe ecran  $n$  caractere "\*" așezate sub forma unui triunghi isoscel (trapez) ca în exemplul următor:

Pentru  $n=16$  se va afișa

```

*
***
*****
*****
*****

```

Pentru  $n=12$  se va afișa

```

**
****
*****

```

Pentru  $n=5$  se va afișa **IMPOSIBIL**.

Cerința problemei este ca primul rând afișat să conțină unul sau cel mult două caractere, iar fiecare linie să difere de precedenta prin exact două caractere.

**Soluție:** problema admite soluție dacă numărul  $n$  poate fi scris ca sumă de numere pare sau ca sumă de numere impare deci trebuie să respecte una din următoarele două forme:  $n=k*k$  sau  $n=k*k+k$ ;

Cazul a)

$n=2+4+6+\dots+2*k=2*(1+2+3+\dots+k)=2*k*(k+1)/2=k*(k+1)=k*k+k$ , unde  $k$  va reprezenta numărul de linii pe care se va face afișarea.

Cazul b)

$n=1+3+\dots+2*k+1=(1+2+3+\dots+2*k+1)-(2+4+6+\dots+2*k)=(2*k+1)*(2*k+2)/2-k*(k+1)=k*k+2*k+1=(k+1)*(k+1)$  unde  $k+1$  reprezintă numărul de linii pe care se va face afișarea.

```

1  var n,k,i,j,l,ll:integer;
2      ok:boolean;
3
4  begin
5      write('N = ');
6      readln(n);
7      k:=trunc(sqrt(n));
8      ok:=false;
9      if k*k=n then begin
10         l:=k-1;
11         ll:=1;
12         ok:=true;
13     end
14     else
15         if k*k+k=n then begin
16             l:=k-1;
17             ll:=2;
18             ok:=true;
19         end;
20     if ok then begin
21         for i:=1 to k do begin
22             for j:=1 to l do
23                 write(' ');
24             for j:=1 to ll do
25                 write('*');
26             writeln;
27             l:=l-1; ll:=ll+2;
28         end;
29     end
30     else writeln('IMPOSIBIL');
31 end.

```

## 1.2.4 Probleme propuse

1. Realizați un program care, pentru un număr natural  $n$  calculează produsul primelor  $n$  numere naturale pare. Algoritmul va fi reprezentat în pseudocod.

**Exemplu:** Pentru  $n=2$  se va afișa 8 adică  $2*4$

2. Realizați un program care citește de la tastatură  $n$  numere naturale și determină media aritmetică a numerelor prime.

**Exemplu:** Pentru  $n=3$  și numerele 4, 5 și 7 se va afișa 6.00

3. Citind de la tastatură  $n$  numere naturale să se calculeze produsul celor care sunt prime cu  $n$ .

**Exemplu:** Pentru  $n=3$  și numerele 4, 5 și 6 se va afișa 20

4. Pentru un număr  $n$  să se afișeze ultimii  $p$  divizorii proprii ai lui (diferiți de 1 și de el însuși). Dacă numărul  $n$  are mai puțin de  $p$  divizori se vor afișa toți.

**Exemplu:** Pentru  $n=24$  și  $p=2$  se va afișa 8 12

5. Realizați un program care calculează media aritmetică a tuturor numerelor palindrom din intervalul  $[a,b]$ .

**Exemplu:** Pentru  $a=8$  și  $b=13$  se va afișa 9.33

6. Realizați un program care permite afișarea primelor  $n$  puteri ale lui 2 care aparțin intervalului  $[x,y]$ .

**Exemplu:** Pentru  $n=2$  și intervalul  $[5,40]$  se va afișa 8 16

7. Realizați un program care permite afișarea valorilor următoarelor expresii, în care  $n$  este un număr natural strict mai mic ca 9, citit de la tastatură.

$E1=12+23+34+\dots+n(n+1)$

$E2=1+12+123+1234+\dots+12\dots n$ .

8. Se consideră un număr  $n$  citit de la tastatură. Realizați un program care să permită afișarea pe ecran a unor caractere dispuse ca în exemplul următor:

**Exemplu:** pentru  $n=5$  se va afișa :

a)	b)	c)	d)
* * * * *	* * * * *	*	1
* * * * *	* * * * *	* *	1 2
* * * * *	* * *	* * *	1 2 3
* * * * *	* *	* * * *	1 2 3 4
* * * * *	*	* * * * *	1 2 3 4 5

9. Scrieți un program care citește de la tastatură  $n$  numere naturale nenule ( $< 32000$ ) și afișează numărul format prin alipirea cifrelor numărului maxim cu cel minim (în această ordine).

*Exemplu:*

Pentru  $n=3$  și numerele 63, 153 și 62 se va afișa 15362

10. Cunoscându-se limitele întregi  $a, b$  ale unui interval și un șir de  $n$  valori, să se realizeze un program care determină media aritmetică a numerelor citite, pentru care suma cifrelor aparține intervalului  $[a, b]$ .

*Exemplu:* Pentru  $a=2, b=10, n=5$  și valorile 11, 39, 32, 80, 84 se va afișa 41.00.

11. Să se realizeze un program care afișează pe ecran toate modalitățile de scriere a valorii  $S$  ca sumă de trei termeni nenuli distincți.

*Exemplu:* Pentru  $S=8$  se va afișa :

$8=1+2+5$

$8=1+3+4$

12. Se citește de la tastatură un șir de  $n$  numere. Realizați un program care verifică dacă numărul format din cifrele unităților acestora este un număr prim.

*Exemplu:* Pentru  $n=4$  și numerele 237 23 453 11 se va afișa : Numărul 7331 este prim.

13. Se citesc de la tastatură  $n$  numere naturale. Afișați numărul de triplete de valori citite consecutiv care pot reprezenta laturile unui triunghi.

*Exemplu:* Pentru  $n=5$  și valorile: 10, 3, 4, 5, 7 se va afișa 2 (3, 4, 5 și 4, 5, 7)

14. Să se afișeze toate tripletele de numere pitagorice mai mici decât un număr  $n$  dat.

15. Se citesc de la tastatură prețurile a  $n$  obiecte achiziționate de o persoană. Valorile citite sunt distincte. Să se afișeze prețurile celor mai scumpe două obiecte cumpărate.

*Exemplu:* Pentru  $n=5$  și valorile 18000, 230, 190000, 2400, și 2000000 se va afișa: 190000 și 2000000

16. Se cunosc notele a  $n$  elevi la un extemporal. Să se afișeze care este nota maximă la test și de către câți elevi a fost obținută.

*Exemplu:*

Pentru  $n=7$  și notele: 4, 6, 4, 8, 8, 5, 8, se va afișa : 'Nota 8 obținută de 3 elevi'.

17. Pentru fiecare elev din cei  $n$  înscriși într-o clasă se cunosc cele două note obținute la educație fizică. Realizați un program care determină media pe clasă la educație fizică, numărul elevilor corigenți și media maximă obținută în clasă. Notele obținute de un elev vor fi citite succesiv.

*Exemplu:*

Pentru  $n=3$  și notele: 8, 6, 8, 10, 8, 8 se va afișa :

'Media clasei este 8, Nici un elev corigent, Media maxima este 9'

18. Citind de la tastatură  $n$  numere naturale, să se calculeze media armonică a lor. Această medie se calculează după formula:

$$\frac{n}{\frac{1}{nr1} + \frac{1}{nr2} + \dots + \frac{1}{nrN}}$$

19. Se consideră un șir de  $n$  perechi de numere naturale care reprezintă limitele întregi ale unor intervale. Citirea intervalelor se face în ordine crescătoare a limitelor inferioare ale intervalelor. Realizați un program care afișează:

- numărul de intervale disjuncte cu primul interval citit.
- numărul de intervale incluse în primul interval citit.

*Exemplu:* Pentru  $n=5$  și intervalele: (2,6) (3,7) (3,5) (10,12) (15,29) se va afișa : 2 1 (două intervale disjuncte cu (2,6) și un interval inclus în (2,6))

20. Să se afișeze cele mai mari două numere prime strict mai mici decât numărul natural  $n$  ( $n > 4$ ).

*Exemplu :* Pentru  $n=19$  se va afișa 13 și 17

21. Se consideră un număr  $n$  citit de la tastatură. Să se realizeze un program care afișează pe ecran cifrele pare ale acestuia în ordinea inversă a apariției, separate prin câte o virgulă.

*Exemplu* Pentru  $n=1234$  se va afișa 4,2

22. Se consideră un număr natural  $n$ . Să se formeze două noi numere, unul format din cifrele pare ale lui  $n$ , celălalt format din cifrele impare.

*Exemplu :* Pentru  $n=13854$  se va afișa 84 și 135

23. Se consideră un număr natural  $n$  ( $n > 1000$ ). Să se afișeze cele două numere formate prin 'înjumătățirea' scrierii zecimale a lui  $n$ .

*Exemplu :* Pentru  $n=12345$  se va afișa 12 și 345. Pentru  $n=182345$  se va afișa 182 și 345.

24. Se consideră un număr natural  $n$  ( $n > 100$ ). Să se afișeze cifrele lui situate pe poziții impare începând cu cifra unităților. *Exemplu :* Pentru  $n=1235$  se va afișa 52

25. Se consideră un număr  $n$  natural ( $n > 100$ ). Să se afișeze suma primelor două cifre ale lui  $n$ .

*Exemplu :* Pentru  $n=1235$  se va afișa 3

26. Se consideră un număr  $n$  natural. Să se afișeze cel mai mic multiplu par al numărului format din prima și ultima cifră a acestuia.

*Exemplu:* Pentru  $n=1235$  se va afișa 30

27. Se consideră un număr natural  $n$  ( $n > 1000$ ). Să se afișeze numărul format din cifrele pare ale lui  $n$  situate pe poziții impare începând cu prima cifră a sa.

*Exemplu:* Pentru  $n=724582$  se va afișa 48

28. Se consideră un număr natural  $n$  ( $n > 1000$ ). Să se afișeze numărul de apariții a cifrei unităților în scrierea lui  $n$ .

*Exemplu:* Pentru  $n=15535$  se va afișa 3 (5 apare de 3 ori)

29. Se consideră un număr natural  $n$  ( $n > 1000$ ). Să se afișeze cea mai mare cifră care apare în scrierea lui  $n$  și numărul de apariții al ei.

*Exemplu:* Pentru  $n=19539$  se va afișa "9 apare de 2 ori"

30. Să se afișeze toate numerele pare începând cu valoarea 2, cât timp suma celor afișate nu este mai mare decât numărul  $n$  natural citit.

*Exemplu:*  $n=15$  se va afișa 2 4 6

31. Se citește de la tastatură un număr  $n$  impar. Să se afișeze primele  $n$  perechi de numere consecutive a căror sumă este divizibilă cu numărul  $n$ .

*Exemplu:* Pentru  $n=3$  se va afișa:

1 2  
4 5  
7 8

32. Se consideră un număr  $n$ . Dacă numărul este palindrom se va afișa numărul format din cifra zecilor și cea a unităților, în caz contrar se va afișa prima cifră a sa. Un număr este palindrom dacă este egal cu numărul obținut cu cifrele citite de la dreapta spre stânga.

*Exemplu:*

Pentru  $n=31413$  se va afișa numărul 13 ;

Pentru numărul 3214 se va afișa numărul 3

33. Realizați un program care citind de la tastatură un număr real, afișează succesiv câte un număr mai mic cu 2 decât ultimul afișat, începând cu valoarea citită. Lista continuă cât timp valorile afișate sunt pozitive.

*Exemplu:* Pentru  $n=7.01$  se va afișa: 7.01 5.01 3.01 1.01

34. Se citește de la tastatură două numere naturale  $n$  și  $m$ . Realizați un program care afișează o listă de numere impare consecutive, cât timp diferența dintre primul și ultimul număr din listă nu este mai mare decât  $m$ . Lista va începe cu primul număr impar mai mare decât  $n$ .

*Exemplu:*

Pentru  $n=7$  și  $m=50$  se va afișa: 9 11 13.....59

35. Realizați un program care, citind de la tastatură un număr  $n$ , afișează puterile lui  $n$  mai mici decât 30000.

36. Realizați un program care afișează cifrele unui număr natural  $x$  în urma conversiei sale în baza  $b$  ( $b \leq 9$ ).

*Exemplu:*

Pentru  $x=128$  și  $b=7$  se va afișa: 242.

37. Se citește de la tastatură două numere. Realizați un program care îl afișează pe cel care are un număr mai mare de cifre de 1 în scrierea în baza 2.

38. Se introduc de la tastatură numere, cât timp ultimul citit nu este egal cu suma precedentelor două. Să se calculeze suma numerelor citite.

*Exemplu:* Pentru valorile 3, 5, 2, 4, 6 se va afișa 20

39. Se citește de la tastatură un număr  $n$ . Să se afișeze pe o singură linie primele  $n$  numere prime.

*Exemplu:* Pentru  $n=4$  se va afișa 2, 3, 5, 7

40. Se consideră un număr  $n$ . Dacă numărul format din primele lui două cifre este perfect, atunci se va afișa numărul total de cifre al lui  $n$ , în caz contrar se va afișa numărul de cifre pare pe care le conține.

Un număr este perfect dacă suma divizorilor săi, strict mai mici, este egală cu numărul respectiv:  $6=1+2+3$ .

*Exemplu:*

Pentru  $n=28413$  se va afișa numărul 5; Pentru numărul 4914 se va afișa numărul 2.

41. Se citește de la tastatură, un șir de  $n$  numere naturale. Realizați un program pentru determinarea numărului din șir cu cei mai mulți divizori.

42. Se citește de la tastatură, un șir de  $n$  numere reale. Realizați un program care determină numărul de apariții al celui mai mare număr prim din șir.

43. Se citește de la tastatură, un șir de  $n$  numere reale. Realizați un program care determină suma maximă a două numere din șir.

*Exemplu:* Pentru  $n=4$  și valorile 8, 3, 8 5 se va afișa 16.

44. Se citește de la tastatură, un șir de  $n$  numere naturale. Realizați un program care determină numărul total de cifre al tuturor numerelor prime din șir.

45. Se consideră un număr natural  $n$ . Să se realizeze un program pentru determinarea numărului perechilor  $(a, b)$  de numere naturale  $(a, b \leq n)$  ce au proprietatea că  $a$  și  $b$  sunt prime între ele.

46. Se consideră un număr natural  $n$ , ( $100 \leq n$ ), format din maxim 9 cifre. Realizați un program care verifică dacă numărul  $n$  este "bine ordonat". Numerele "bine ordonate" sunt cele care au proprietatea că cifrele lor apar fie în ordine crescătoare, fie descrescătoare.

47. Să se alcătuiască algoritmul care permite afișarea factorilor primi și a puterilor la care aceștia apar în descompunerea unui număr natural  $n$ , dat.

48. Se introduc de la tastatură  $n$  numere întregi. Afișați pe ecran câte dintre ele au suma cifrelor egală cu numărul de ordine avut la citire.

*Exemplu:*

Pentru  $n=5$  și numerele 2, 101, 23, 3001, 234 pe ecran se va afișa 2 (101 și 3001).

49. Se consideră trei numere naturale  $n$ ,  $a$  și  $b$  ( $a \leq b \leq n$ ). Să se creeze un program care permite afișarea factorilor primi ce aparțin intervalului  $(a, b)$  și a puterilor la care aceștia apar în descompunerea lui  $n$ .

*Exemplu:* pentru  $n=36$ ,  $a=1$ ,  $b=10$  se va afișa:

2 exponent 2

3 exponent 2

50. Se citește de la tastatură un șir de  $k$  intervale, pentru fiecare fiind introduse cele două limite  $[a, b]$  unde  $a < b$  numere întregi. Alcătuiți un program care să permită afișarea limitelor intervalului ce conține cele mai multe valori întregi.

*Exemplu:*

Pentru  $k=4$  și șirul de numere: 2 6 4 8 3 22 5 10 se va afișa intervalul (3 22).

51. Creați un program care să simuleze următorul joc: doi parteneri introduc alternativ câte un caracter de la tastatură. Pierde acel jucător care a introdus un caracter identic cu ultimul introdus de el sau de partenerul său.

52. Afișați primele  $n$  perechi de numere prime consecutive pe mulțimea numerelor impare. *Exemplu:* pentru  $n=3$  se va afișa (3,5), (5,7) (11,13).

53. Se dau două numere naturale nenule  $a$  și  $n$  cu cel mult 10 cifre. Se cere să se afișeze ultima cifră a puterii  $a^n$ .

*Exemplu:* Pentru  $a=1222223$  și  $n=180$  se va afișa 1.

54. Pentru  $n$  număr natural nenul cu cel mult 5 cifre se cere să se determine în câte cifre de 0 se termină produsul  $1 \cdot 2 \cdot \dots \cdot n$ .

*Exemplu:* pentru  $n=16$  se va afișa 3

55. Afișați toate numerele naturale nenule de la 1 la  $n^2$  în ordine crescătoare, câte  $n$  pe un rând (separate prin câte un spațiu).

*Exemplu:* Pentru  $n=4$  se va afișa

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

56. Se dau două numere naturale nenule cu cel mult 10 cifre. Se cere să se afișeze cifrele comune.

*Exemplu:*

Pentru numerele 12323234 și 657284 se va afișa 2 4.

57. Fie două cifre zecimale  $a$  și  $b$ . De la tastatură se va introduce un șir de numere naturale, citirea terminându-se cu o dată cu introducerea valorii 0 (care nu va face parte din șir). Să se afișeze numărul de valori citite cu proprietatea că au în scrierea zecimală succesiunea de cifre  $ab$ .

*Exemplu:*

Pentru  $a=2$ ,  $b=4$  și valorile 2342, 420, 8248, 15264, 24245, 0 se va afișa 2

58. Fiind dat un număr natural de maxim 9 cifre, să se determine între care din cifre se poate plasa operatorul de înmulțire astfel încât produsul celor două numere obținute să fie maxim.

*Exemplu:*

Pentru valoarea 3203 se va afișa  $320 \cdot 3 = 960$ .

59. Se citesc de la tastatură mai multe șiruri de numere întregi, fiecare terminându-se cu valoarea 0. Operația de citire se încheie la introducerea valorii 0 de două ori consecutiv. Determinați cel mai mare număr prim din fiecare șir sau -1 dacă nu există număr prim.

*Exemplu:*

Pentru șirul 3 5 63 0 5 6 9 11 0 33 0 17 0 13 15 0 3 4 5 6 7 0 5 0 0, se va afișa a 5, 11, -1, 17, 13, 7, 5

60. Să se afișeze primele  $n$  numere naturale care au cifra de control egală cu cifra  $x$  citită de la tastatură. Cifra de control a unui număr se obține prin însumarea cifrelor din scrierea zecimală a numărului, apoi se însumează cifrele acestei sume, ș.a.m.d până când suma obținută este exprimată printr-o cifră. De exemplu numărul 28997 are cifra de control 9 ( $2+8+9+9+7=35$ ,  $3+5=8$ ).

*Exemplu:*

Pentru  $n=2$  și  $x=3$  se va afișa: 3, 12



### 1.3. Probleme de concurs ce procesează date simple

#### 1.3.1 Probleme rezolvate

1. (Pomi - \*) Se consideră un șir de  $n < 100.000$  copaci numerotați cu numerele de la 1 la  $n$ . Un copil mai năzdravan concepe un joc după următoarea regulă. În primul minut al jocului copilul se ascunde după copacul 1, la minutul doi în spatele copacului 2, ș.a.m.d. Sensul de deplasare al copilului se schimbă dacă minutul în care se află este un număr divizibil cu 6 sau are ultima cifră 7.

De asemenea, jocul încetează dacă năzdravanul nostru este pus să iasă în afara șirului de  $n$  pomi, adică respectând regula de deplasare copilul ar trebui să se ascundă în spatele copacului 0 sau  $n+1$ .

Minut	Pom	Minut	Pom	Minut	Pom	Minut	Pom
1	1	6	6	11	9	16	6
2	2	7	5	12	10	17	5
3	3	8	6	13	9	18	6
4	4	9	7	14	8	19	5
5	5	10	8	15	7	20	4

Să se determine care este copacul în spatele căruia se ascunde copilul la momentul  $k < 100.000$ . Dacă jocul a fost întrerupt înainte sau la momentul  $k$ , atunci se va scrie valoarea 0.

Exemplu:  $n=10$   $k=20$ .

Soluție: Se vor simula mișcările copilului respectând regulile impuse în enunț.

```
1  intreg n, k;  
2  citeste n, k;  
3  poz ← 0;  
4  sens ← 1;  
5  pentru i ← 1, k executa  
6      poz ← poz + sens;  
7      dacă (i mod 6=0) sau (i mod 10=7) atunci  
8          sens ← sens * -1;  
9      dacă (poz<1) sau (poz>n) atunci  
10         scrie 0; stop.  
11  
12 scrie poz;  
13 stop.
```

2. (Număr de valori divizibile cu 3 -\*\*) Fie șirul de numere 1, 12, 123, 1234, ...12345678910,.... Considerându-se primele  $n$  numere ale acestui șir, trebuie determinat numărul de numere care sunt divizibile cu 3.

Exemplu: Pentru  $n=5$  se va afișa 4.

Soluție:

Se caută o regulă pe care o respectă așezarea numerelor divizibile cu 3 în șirul dat, și se observă că atunci când restul lui  $n$  este 0 sau 1 răspunsul este  $2*(n \text{ div } 3)$ , iar când restul este 2 răspunsul este  $2*(n \text{ div } 3)+1$ .

```
1  intreg n;  
2  citeste n;  
3  dacă (n mod 3=0) sau (n mod 3=1) atunci  
4      scrie 2*(n div 3)  
5  altfel  
6      scrie 2*(n div 3)+1;  
7  
8  stop.
```

3. (987654321 - \*\*) Pentru un număr  $n$  dat, trebuie să realizați un program care afișează câte numere de  $n$  cifre din baza 10 au proprietatea că pătratul lor se termină în secvența de cifre: 987654321.

Numărul  $n$  se va citi de la tastatură și va avea maxim 6 cifre.

Soluție: Observația de bază este că ultimele 9 cifre, care ne interesează, ale unui pătrat sunt influențate doar de ultimele nouă cifre ale numărului ce va fi ridicat la pătrat. Astfel, după o simplă căutare, pentru  $n=9$  există 8 numere care ridicate la pătrat se termină în 987654321 (111111111, 888888889, etc.) Astfel pentru  $n>9$  răspunsul va fi  $8*9*10^{n-10}$  (prima cifră trebuie să fie diferită de 0).

```
1  intreg n, i;  
2  citeste n;  
3  dacă n < 9 atunci scrie 0  
4  altfel  
5      dacă n = 9 atunci scrie 8  
6      altfel  
7          scrie 72  
8          pentru i ← 1, n-10 executa  
9              scrie 0  
10  
11  
12
```

4. (Numere aproape prime - \*\*) Definim un număr "aproape prim" un întreg pozitiv care poate fi scris ca produsul a două numere prime. Dându-se o secvență de  $n < 11$  întregi pozitivi de cel mult 9 cifre, realizați un program care afișează pentru fiecare dintre ele mesajul "DA" dacă numărul este "aproape prim" și "NU" în caz contrar.

Exemplu:

n=2	DA
6	NU
8	

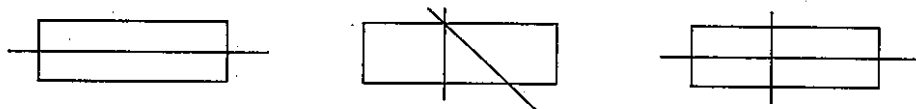
**Soluție:** Se factorizează fiecare număr citit și se numără câți factori primi are și se afișează mesajul corespunzător.

```

1  întreg n, i, nr, c;
2  citește n;
3  pentru i ← 1, n executa
4      citește nr; c ← 0;
5      pentru d ← 2, √nr executa
6          dacă nr mod d = 0 atunci
7              c ← c + 1;
8              cat timp nr mod d = 0 executa
9                  nr ← nr div d;
10             ■
11         ■
12     ■
13     dacă nr > 1 atunci c ← c + 1;
14     ■
15     dacă c = 2 atunci scrie "DA"
16     altfel scrie "NU"
17     ■
18     ■

```

5. (Trasarea liniilor - \*\*\*) Se consideră o foaie de hârtie pe care Gigel va desena linii. Trasând o singură linie el împarte foaia de hârtie în două zone. Dacă va trasa 2 linii el poate obține 2 (linii suprapuse), 3 sau chiar 4 zone.



Determinați care este numărul maxim de zone în care poate fi împărțită foaia de hârtie trasând  $n$  linii ( $0 < n < 35000$ ).

Exemplu: Pentru  $n=3$  se va afișa 7

**Soluție:** Dacă am trasat  $k$  linii, putem trasa a  $k+1$ -a linie, astfel încât să intersecteze toate cele  $k$  linii trasate anterior, astfel formându-se încă  $k+1$  zone. Astfel, formula pentru numărul de zone va fi  $1+1+2+3+\dots+n=1+n*(n+1)/2$ .

```

1  întreg n;
2  citește n;
3  scrie 1 + n*(n + 1) div 2;
4  stop.

```

6. (Numărătoare - \*\*\*) Determinați numărul de valori dintr-o secvență de  $N$  numere naturale cel mult egale cu 10.000, care după ridicare la puterea  $M$  se vor divide cu  $K$ . ( $0 < N, M, K < 10.001$ )

Exemplu:

N=4 M=2 K=50	1
9 10 11 12	

**Soluție:** Fiecare număr este ridicat la puterea  $M$  păstrând la fiecare pas restul împărțirii la  $K$ , nu întregul număr. Pentru a face ridicarea la putere eficient ne folosim de relațiile:  $a^{2p}=(a^p)^2$  și  $a^{2p+1}=a*a^{2p}$ .

```

1  întreg n, m, k, i, j, nr, b, aux, rez;
2  citește n, m, k;
3  rez ← 0;
4  pentru i ← 1, n executa
5      citește nr;
6      aux ← nr; b ← 0;
7      cat timp aux > 0 executa
8          aux ← aux div 2;
9          b ← b + 1;
10         ■
11     aux ← 1;
12     pentru j ← b-1, 0, -1 executa
13         aux ← (aux*aux) mod k;
14         dacă bitul j al lui nr = 1 atunci
15             aux ← (aux*nr) mod k;
16         ■
17     ■
18     dacă aux=0 atunci
19         rez ← rez + 1;
20     ■
21     ■
22 scrie rez; stop.

```

7. (Trepte - \*\*\*) O persoană trebuie să coboare o scară cu  $N < 40$  trepte numerotate 1, 2, ...  $N$ . Știind că la fiecare pas persoana poate să coboare una sau două trepte determinați în câte moduri poate coborî scara. Exemplu:

N=3	2
-----	---

**Soluție:** Se observă că numărul de moduri în care se urcă o scară cu  $N$  trepte este egal cu numărul de moduri în care se urcă o scară cu  $N-1$  trepte (pas de 1) adunat cu numărul de moduri în care se urcă o scară cu  $N-2$  trepte. În concluzie, răspunsul este cel de-al  $N$ -lea termen al șirului Fibonacci.

```

1  intreg n,a,b,c,i;
2  citește n;
3  a ← 1; b ← 1;
4  dacă n = 1 atunci scrie a;
5  ┌
6  │ dacă n = 2 atunci scrie b;
7  │ ┌
8  │ │ pentru i ← 3, n executa
9  │ │ │ c ← a + b;
10 │ │ │ a ← b;
11 │ │ │ b ← c;
12 │ └
13 scrie c; stop.

```

8. (**Majoritar** - \*\*\*) La alegeri au participat  $N < 1.000.000$  alegători fiecare exprimându-și votul sub forma unui număr care reprezintă codul candidatului ales. Să se determine dacă există un candidat care este majoritar, adică numărul de apariții ale codului său este mai mare decât  $N/2$ . Se presupune că un astfel de candidat există. *Exemplu:*

N=7  
1 1 3 1 2 1 1

1

**Soluție:** Vom parcurge șirul o singură dată și vom reține o variabilă  $c$  pentru candidatul considerat curent majoritar și  $nr$  numărul de apariții necontracate ale candidatului. La început considerăm că primul element din șir este majoritar. La fiecare pas, verificăm dacă numărul curent este egal cu candidatul sau nu și modificăm numărul de apariții în funcție de asta. Dacă numărul de apariții ajunge la un moment dat 0, cu siguranță nu am găsit candidatul majoritar și schimbăm valoarea variabilei  $c$  cu numărul curent citit.

```

1  intreg n, i, c, nr;
2  citește n;
3  c ← -1; nr ← 1;
4  pentru i ← 1, n executa
5  │ citește x;
6  │ │ dacă x = c atunci
7  │ │ │ nr ← nr + 1;
8  │ │ │ altfel
9  │ │ │ nr ← nr - 1;
10 │ │ └
11 │ │ │ dacă nr = 0 atunci
12 │ │ │ │ c ← x; nr ← 1;
13 │ └
14 └
15 scrie c;
16 stop.

```

9. (**Virus** - \*\*\*) La laboratorul de cercetări genetice din Lugoj, în urma unor experiențe nereușite, un virus a suferit mutații. Ca urmare, există trei tipuri de viruși: virusul inițial (îl vom numi virus de tip  $A$ ) și două tipuri de viruși mutați (îi vom numi viruși de tip  $B$  și viruși de tip  $C$ ). Pentru a studia comportamentul virușilor, cercetătorii au izolat într-un mediu steril un virus de tip  $A$ , un virus de tip  $B$  și un virus de tip  $C$  și au observat că la fiecare secundă se produc următoarele transformări:

- orice virus de tip  $A$  se divide și se obțin un nou virus de tip  $A$ , un nou virus de tip  $B$  și un nou virus de tip  $C$ .
- dintr-un virus de tip  $B$  și un virus de tip  $C$  se obțin trei noi viruși (unul de tip  $C$  și doi de tip  $B$ ).

De asemenea, cercetătorii au observat că virușii nu mor.

Să se determine numărul de viruși de fiecare tip, existenți după  $N < 26$  secunde.

*Exemplu:*

N=2

4A 12B 8C

(Concurs "Grigore Moisil", Lugoj 2001, cls. V-VI)

**Soluție:** Se simulează evoluția populației de viruși timp de  $n$  secunde, determinând la fiecare moment câți viruși există din fiecare tip.

```

1  intreg n,i,a,b,c;
2  citește n;
3  a ← 1; b ← 1; c ← 1;
4  pentru i ← 1, n executa
5  │ b ← b + 2*c + a;
6  │ c ← 2*c + a;
7  │ a ← 2*a;
8  └
9  scrie a,b,c; stop.

```

10. (**Broasca** - \*\*) O broască se deplasează efectuând câte o săritură de lungime  $p$  cm la fiecare secundă. După fiecare  $n$  secunde broșcuța devine mai obosită, iar lungimea săriturii pe care o face se înjumătățește. Scrieți un program care citește numerele naturale mai mici ca 30.000,  $p$ ,  $n$ , și  $T$  – durată totală a deplasării broșcuței exprimată în secunde și care afișează distanța totală pe care a parcurs-o broșcuța cu doua zecimale. Se știe că  $T/n < 16$ .

*Exemplu:*

n=10  
p=20  
T=33

35750.00

(Olimpiada Națională de Informatică Gimnaziu, Gălăciuc 2001, cls. V)

**Soluție:** Se simulează mișcarea broaștei pas cu pas, adunând la fiecare  $n$  secunde numărul de centimetri parcurși de broșcuță.

```

1  integ n, t;
2  real d, p;
3  citește n, p, t;
4  d ← 0;
5  cat timp n ≤ t executa
6  |   d ← d + n*p;
7  |   p ← p*0.5;
8  |   t ← t - n;
9  |
10 d ← d + t*p;
11 scrie d;

```

### 1.3.2 Probleme propuse

1. (**Cercetași - \*\***) În zona Vrancea există două grupuri de cercetași, cu sediul în două regiuni (Gălăciuc și Soveja). Cercetașii din Gălăciuc sunt organizați în *D1* detașamente a câte *N1* cercetași fiecare. Cercetașii din Soveja sunt organizați în *D2* detașamente a câte *N2* cercetași fiecare. În urma unui ordin de la Organizația Europeană a Cercetașilor, trebuie ca toate detașamentele din zona Vrancea să fie formate din același număr de cercetași.

Pentru îndeplinirea ordinului, trebuie reorganizate detașamentele din fiecare regiune, fără a deplasa cercetași dintr-o regiune în alta.

Scrieți un program care citește *D1*, *N1*, *D2*, *N2* ( $< 1000$ ) și care să afișeze numărul de cercetași din fiecare detașament după reorganizare, cât mai mare posibil și câte detașamente se formează în zona Gălăciuc și câte detașamente se formează în zona Soveja.

Exemplu:

D1=3	Nr. de cercetasi din fiecare
N1=15	detasament:9
D2=2	Nr. de detasamente din Galaciuc:5
N2=18	Nr. de detasamente din Soveja:4

(Olimpiada Națională de Informatică Gimnaziu, Gălăciuc 2001, cls. V)

2. (**La școală - \*\***) Directorul unei școli dorește să premieze la sfârșitul anului școlar pe cei mai buni elevi la învățătură. Pentru acest lucru el are de rezolvat două probleme:

a) Să determine câți elevi vor fi premiați dintre cei  $n$  ( $2 \leq n \leq 700$ ) elevi ai școlii. După discuții aprinse cu ceilalți profesori se hotărăște în Consiliul Profesoral ca numărul premianților să fie  $n-k$ , unde  $k$  este cel mai mare număr pătrat perfect mai mic strict decât  $n$ . De exemplu, pentru  $n=150$ ,  $k$  este 144 (pentru că  $144=12^2$ ), deci vor fi premiați  $150-144=6$  elevi.

b) Pentru a fi cât mai multă liniște la premiere, în Consiliul Profesoral se ia decizia ca elevii care nu vor fi premiați să fie așezați pe terenul de sport pe rânduri de câte  $p$  elevi (unde  $p^2=k$ ). În acest scop, directorul a numerotat elevii nepremiați de la 1 la  $k$  și a hotărât ca elevii să fie așezați în ordinea descrescătoare a numerelor asociate.

Scrieți un program care:

- citește de la tastatură  $n$ , numărul de elevi din școală;
- determină și afișează pe ecran numărul de elevi premiați;
- afișează pe ecran modul de așezare a elevilor nepremiați

Exemplu:

$n=35$

```

Numarul de elevi premiați: 10
Elevii nepremiați:
25 24 23 22 21
20 19 18 17 16
15 14 13 12 11
10 9 8 7 6
5 4 3 2 1

```

(Olimpiada Județeană de Informatică Gimnaziu, 2002, cls. V)

3. (**Șir - \*\***) Se consideră următorul șir de numere naturale:

7, 17, 37, 47, 67, 97, 107, 127, 137, 157, 167, ...

Deduceți regula după care sunt generați termenii șirului și afișați pe ecran al  $N$ -lea ( $N < 2001$ ) termen din șirul de mai sus.

Exemplu:

$N=10$

| 157

(Concurs "Grigore Moisil", Lugoj 2001, cls. V-VI)

4. (**Balaur - \*\***) A fost o dată un balaur cu 6 capete. Într-o zi Făt-Frumos s-a supărat și i-a tăiat un cap. Peste noapte i-au crescut alte 6 capete în loc. Pe același gât! A doua zi, Făt-Frumos iar i-a tăiat un cap, dar peste noapte balaurului i-au crescut în loc alte 6 capete ... și tot așa timp de  $n$  zile. În cea de a  $(n+1)$ -a zi, Făt-Frumos s-a plictisit și a plecat acasă!

Scrieți un program care citește de la tastatură  $n$ , numărul de zile, și care afișează pe ecran câte capete avea balaurul după  $n$  zile.

Exemplu:

$n=3$

| Balaurul are 15 capete

(Olimpiada Județeană de Informatică Gimnaziu, 2002, cls. V)

5. (**Valori-pantă - \*\***) Se dă un șir de  $N$  ( $1 \leq N \leq 30$ ) elemente numere naturale (cu maxim 8 cifre). Se cere:

a) Să se afișeze câte elemente din șir sunt valori-pantă (numere care privesc de la stânga sau de la dreapta au cifrele în ordine crescătoare) De exemplu, 136 și 931 sunt valori-pantă.

b) Să se afișeze cea mai mare și cea mai mică valoare-pantă.

Dacă la punctul a) sunt 0 valori-pantă, atunci la b) se va afișa mesajul NU EXISTA

Exemplu:

N=6  
126  
9621  
1212  
3678  
9231  
9621

Numarul de valori-panta: 4  
Cea mai mare valoare-panta: 9621  
Cea mai mica valoare-panta: 126

(Olimpiada Județeană de Informatică Gimnaziu, 2002, cls. VI)

6. (Gigel - \*\*) Gigel este un tip ciudat. Lui îi place să își impresioneze colegii exprimând duratele numai în secunde. De exemplu, dacă îl vei întreba cât e ceasul el îți va răspunde câte secunde s-au scurs de la ora 0.00 din ziua respectivă. Dacă ai să-l întrebi ce vârstă are, el îți va răspunde câte secunde au trecut de când s-a născut.

Colegii lui Gigel au hotărât că nu e cazul să se lase impresionați; ca urmare au nevoie de un program care să citească de la tastatură un număr natural  $N$  ( $N \leq 2000000000$ ) care reprezintă vârsta lui Gigel exprimată în secunde și care va afișa pe ecran câți ani, câte luni și câte zile are Gigel (orele și minutele rămase sunt considerate nesemnificative). Scrieți acest program pentru colegii lui Gigel!

Nu uitați că anii bisecți sunt cei divizibili cu 4, dar nedivizibili cu 100 sau divizibili cu 400. De exemplu 1992 și 2000 au fost ani bisecți. Dar anul 1900 nu a fost bisect. Anii bisecți au 366 de zile, spre deosebire de ceilalți care au doar 365.

Considerăm că ne aflăm în ultima zi de școală (15 iunie 2002).

Exemplu:

N=69206400

Gigel are 2 ani, 2 luni și 10 zile

(Olimpiada Națională de Informatică Gimnaziu, Gălăciuc 2002, cls. V)

7. (Exponent - \*\*\*) Se dă un număr natural  $n < 101$  și o cifră  $k$  din mulțimea  $\{2, 3, 5, 7\}$ . Se cere să se afișeze exponentul lui  $k$  în descompunerea în factori primi a produsului  $1 \cdot 2 \cdot 3 \cdot \dots \cdot n$ .

Exemplu:

n=6  
k=3

2

(Olimpiada Județeană de Informatică Gimnaziu, 2003, cls. V)

8. (Pinocchio - \*\*) În fiecare zi nelucrătoare din săptămână, Pinocchio spune câte o minciună datorită căreia nasul acestuia crește cu câte  $p < 101$  centimetri pe zi. Sâmbăta și duminică, când vine bunicul Gepeto acasă, pentru a nu-l supăra prea tare, Pinocchio reușește să nu spună nici o minciună, ba chiar uitându-se în oglindă observă că în fiecare din aceste zile lungimea nasului său scade cu câte 1 centimetru pe zi. Când începe o nouă săptămână, rămânând singur acasă Pinocchio continuă șirul minciunilor.

Care este dimensiunea nasului lui Pinocchio după  $k < 257$  zile știind că inițial nasul său măsura  $n < 1001$  centimetri?

Exemplu:

n=2  
p=1  
k=8

6 cm

(Olimpiada Județeană de Informatică Gimnaziu, 2003, cls. V)

9. (Gardul - \*\*) Doi copii vopsesc un gard alcătuit din  $n < 100.001$  scânduri pe care le vom numerota de la 1 la  $n$  astfel: primul ia o cutie de vopsea roșie cu care vopsește scândurile cu numărul  $p, 2p, 3p$ , etc. Al doilea procedează la fel, începe de la același capăt al gardului, dar ia o cutie de vopsea albastră și vopsește din  $q$  în  $q$  scânduri.

Astfel, când vor termina de vopsit, gardul va avea multe scânduri nevopsite, unele scânduri vopsite în roșu, altele în albastru, iar altele în violet (cele care au fost vopsite și cu roșu și cu albastru).

Cunoscând numerele  $n, p$  și  $q$  afișați:

- câte scânduri rămân nevopsite;
- câte scânduri sunt vopsite în roșu;
- câte scânduri sunt vopsite în albastru;
- câte scânduri sunt vopsite în violet.

Exemplu:

n=25  
p=4  
q=6

a) 17  
b) 4  
c) 2  
d) 2

(Olimpiada Județeană de Informatică Gimnaziu, 2003, cls. VI)

10. (Săritura cangurului - \*\*) A fost odată ca niciodată, a fost un cangur care creștea ca Făt Frumos din poveste, într-un an precum alții în zece. Într-o zi a început să facă sărituri. Și a sărit pentru început 7 metri. A doua zi a sărit, în plus față de ziua precedentă, de zece ori mai mult. În a treia zi a reușit să sară, în plus față de prima zi, de zece ori mai mult decât în ziua a doua. În a patra zi a sărit, în plus față de prima zi, de zece ori mai mult decât în ziua a treia. Și tot așa mai departe.

Scrieți un program care calculează câți metri a sărit cangurul, în total, în  $n < 12$  zile.

Exemplu:

n=3

861 m

(Olimpiada Națională de Informatică Gimnaziu, Focșani 2003, cls. V)

11. (Cifre - \*\*) Se dau două numere naturale  $a, b$  cu maxim 9 cifre.

- Să se determine cifrele distincte, comune numerelor  $a$  și  $b$ .
- Să se afișeze numărul cel mai mare format din toate cifrele lui  $a$  și  $b$ .

Exemplu:

a=2115  
b=29025

a) 2 5  
b) 955222110

(Olimpiada Județeană de Informatică Gimnaziu, 2004, cls. V)

12. (Concurs - \*\*) La un concurs de matematică participă elevi din mai multe școli din diferite orașe. Pentru a se putea deosebi între ele lucrările lor, fiecare lucrare este codificată printr-un număr natural cu 3 cifre, să zicem  $abc$ , unde  $a$  (cifra sutelor) este codul orașului,  $b$

(cifra zecilor) este codul școlii din orașul  $a$ , iar  $c$  (cifra unităților) este codul unui elev din școala  $b$  din orașul  $a$ .

Exemplu: lucrarea cu codul 328 este lucrarea elevului cu codul 8 de la școala cu codul 2 din orașul cu codul 3. Se cunosc: un cod (al lucrării unui elev  $H$ , prietenul nostru), numărul  $n < 20$  de lucrări premiate și codurile acestora. Codul de oraș (cifra sutelor din fiecare cod) este de la 1 la 5, inclusiv. Codurile școlilor din fiecare oraș (cifra zecilor) este de la 0 la 9, inclusiv, iar codul elevilor (cifra unităților) este tot de la 0 la 9 inclusiv.

Se cere să se rezolve cerințele:

a) Verificați dacă  $H$  este premiant, sau nu.

b) Determinați numărul de premii luate de elevii din orașul lui  $H$  (inclusiv  $H$ , dacă a fost premiat).

c) Determinați numărul de premii luate de elevii din școala lui  $H$  (inclusiv  $H$ , dacă a fost premiat).

Exemplu:

$H=234$

$n=6$

123

232

125

222

421

235

a) NU

b) 3

c) 2

(Olimpiada Județeană de Informatică Gimnaziu, 2004, cls. V)

13. (Vânătoare - \*\*) Vânătorul șef al regelui Arthur a primit însărcinare să vâneze primele rațe ce se întorc din țările calde. Regele fiind un tip cu idei fixe, i-a cerut vânătorului să vâneze rațele albe cu săgeți albe, iar rațele negre cu săgeți negre. Rațele vin în rânduri (stoluri) din ce în ce mai mari: mai întâi una, apoi două, trei, cinci, opt, treisprezece, ș.a.m.d. Se observă că numărul de rațe dintr-un rând este egal cu numărul de rațe de pe cele două rânduri anterioare.

Rațele fiind niște creaturi ordonate zboară în rânduri, în care nu vei putea găsi două rațe de aceeași culoare alăturate, fiecare rând începând cu o rață albă. Vânătorul știe că dacă a început să doboare o rață, trebuie să le doboare pe toate de pe rândul acesteia, deoarece supraviețuitoarele vor alerta celelalte rațe și ele nu se vor mai întoarce niciodată, iar vânătorul nostru își va pierde slujba. Știind că vânătorul a primit  $ka$  săgeți albe și  $kb$  ( $ka, kb < 2.000.000.001$ ) săgeți negre, trebuie să determinați câte rânduri de rațe a doborât și câte săgeți de fiecare tip i-au rămas, știind că el vrea să-și păstreze slujba. Se va afișa pe ecran:

- numărul de rânduri doborâte
- numărul de săgeți albe rămase
- numărul de săgeți negre rămase.

Exemplu:

$ka=9$

$kb=10$

4  
2  
6

(Olimpiada Județeană de Informatică Gimnaziu, 2004, cls. VI)

14. (Excursie - \*\*) Drumul de la Gălăciuc la Soveja este marcat prin  $n+1$  borne succesive, borne numerotate de la 0 la  $n$ . Borna 0 (Gălăciuc) este la altitudinea 0. Pe fiecare dintre următoarele  $n$  borne sunt scrise câte doua numere naturale, primul reprezentând altitudinea locului și al doilea reprezentând distanța față de borna precedentă.

Se consideră că dacă o bornă este la altitudinea  $x$ , iar borna următoare la altitudinea  $y$  ( $x < y$ ), atunci drumul între cele două borne urcă. Dacă o bornă este la o altitudine  $x$ , iar borna următoare este la altitudinea  $y$  ( $x > y$ ), atunci drumul între cele două borne coboară. Dacă o bornă este la altitudinea  $x$ , iar borna următoare tot la altitudinea  $x$ , atunci drumul între cele două borne este tot timpul plat.

Se citește valoarea  $n$  și apoi  $n$  perechi de numere naturale reprezentând valorile înscrise pe cele  $n$  borne. Deoarece turistul care pleacă de la Gălăciuc spre Soveja, găfăie de câte ori urcă, vi se cere să afișați lungimea celei mai lungi porțiuni continue pe care acesta o parcurge fără să găfăie. Dacă există numai porțiuni de urcare, se va afișa valoarea 0. Toate numerele din problemă sunt numere naturale nenule de cel mult două cifre.

Exemplu:

$n=5$

$h1=14$   $d1=3$

$h2=14$   $d2=19$

$h3=8$   $d3=4$

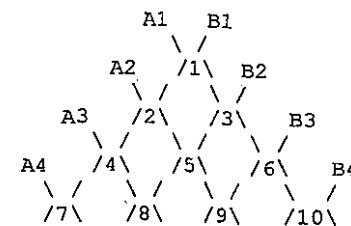
$h4=10$   $d4=10$

$h5=5$   $d5=17$

23

(Olimpiada Națională de Informatică Gimnaziu, Gălăciuc 2001, cls. VI)

15. (Coordonate - \*\*) Numerele naturale se așează într-un triunghi, ca în figura:



Fiecare număr este așezat în vârful unui romb și este unic determinat de cele două diagonale ( $A$  și  $B$ ). Diagonalele sunt numerotate de sus în jos, în acest fel:

- numărul 1 este așezat pe diagonala  $A1$  și diagonala  $B1$
- numărul 2 este așezat pe diagonala  $A2$  și diagonala  $B1$
- numărul 3 este așezat pe diagonala  $A1$  și diagonala  $B2$ , ș.a.m.d.

Fiind dat de la tastatură un număr întreg pozitiv  $n < 32768$ , să se scrie coordonatele celor două diagonale pe care se găsește  $n$ .

Exemplu:

$n=12$   
 $n=6$

A4	B2
A1	B3

(Concurs Sinaia, 1998, cls. VII)

16. (Becuri - \*\*\*) Pentru iluminatul public pe strada Info se găsesc un număr par de stâlpi, pe fiecare stâlp fiind plasat un singur bec, de culoare albă sau galbenă. Pentru a se putea ști și ziua, când becurile sunt stinse, dacă un bec este alb sau galben, elevii de la o școală au scris pe fiecare stâlp un număr. Numărul de pe stâlpii cu număr de ordine impar reprezintă numărul de becuri albe ce sunt în partea stângă a stâlpului (exclusiv stâlpul respectiv).

Numărul de pe stâlpii cu număr de ordine par reprezintă numărul de becuri galbene ce sunt în partea stângă a stâlpului (exclusiv stâlpul respectiv). Un elev mai zburdalnic a modificat numerele de pe stâlpi, astfel încât numărul scris sub fiecare bec galben este eronat, iar numărul scris sub fiecare bec alb este corect. Elevii care învață Informatică au spus că, în aceste condiții, ei pot să determine cu exactitate culoarea fiecărui bec. Scrieți un program care să citească numărul de stâlpi  $N < 1001$  precum și numerele scrise pe cei  $N$  stâlpi și care să afișeze pe ecran culoarea celor  $N$  becuri.

Exemplu:

$N=6$

0	Becul 1: alb
0	Becul 2: alb
1	Becul 3: galben
1	Becul 4: alb
4	Becul 5: galben
2	Becul 6: alb

(Olimpiada Județeană de Informatică, Timiș 2000, cls. IX)

17. (Frații - \*\*\*\*) O proprietate interesantă a fracțiilor ireductibile este că orice fracție se poate obține după următoarele reguli:

- pe primul nivel se află fracția  $1/1$ ;
- pe al doilea nivel, în stânga fracției  $1/1$  de pe primul nivel, plasăm fracția  $1/2$  iar în dreapta ei fracția  $2/1$ ;

nivelul 1:  $1/1$   
 nivelul 2:  $1/2$   $2/1$

- pe fiecare nivel  $k$  se plasează sub fiecare fracție  $i/j$  de pe nivelul de deasupra, fracția  $i/(i+j)$  în stânga, iar fracția  $(i+j)/j$  în dreapta.

nivelul 1:  $1/1$   
 nivelul 2:  $1/2$   $2/1$   
 nivelul 3:  $1/3$   $3/2$   $2/3$   $3/1$

Dându-se o fracție oarecare prin numărătorul și numitorul său, determinați numărul nivelului pe care se află fracția sau o fracție echivalentă (având aceeași valoare) cu aceasta. Se citesc două numere naturale nenule  $N, M \leq 2000000000$ , separate printr-un spațiu, reprezentând numărătorul și numitorul unei fracții, și se va afișa număr natural nenul, reprezentând numărul nivelului care corespunde fracției.

Exemplu:

$N=13$   $M=8$   
 $N=12$   $M=8$

$6$   
 $3$

(Olimpiada Națională de Informatică, Bacău 2001, cls. IX)

18. (Tablou - \*\*\*\*\*) Generați un tablou bidimensional cu proprietățile:

- conține  $N$  linii și  $N$  coloane ( $N < 101$ );
- elementele sale sunt numere naturale nenule;
- suma elementelor este egală cu numărul natural nenul  $S$  ( $S < 2^{31}$ );
- pe nici o linie și pe nici o coloană nu există două elemente identice;
- diferența dintre cel mai mare și cel mai mic element ale tabloului este minimă.

Se citesc două numere naturale nenule, separate printr-un spațiu, reprezentând numărul de linii și de coloane ale tabloului, respectiv valoarea sumei tuturor elementelor din tablou și se va afișa pe  $N$  linii elementele tabloului, câte o linie din tablou; elementele se vor separa prin câte un spațiu. Dacă problema nu are soluție, se va scrie cifra 0. Dacă problema are mai multe soluții, se va scrie una singură.

Exemplu:

$N=3$   $S=51$

$4$   $6$   $7$   
 $7$   $4$   $5$   
 $5$   $7$   $6$

(Olimpiada Națională de Informatică, Bacău 2001, cls. IX)

19. (Ciupercuțe - \*\*) Un vrăjitor bătrân vrea să prepare o licoare specială. Pentru o doză de licoare el are nevoie de  $M$  ciuperci fermecate. O ciupercă este fermecată dacă numărul bulinelor de pe pălăria ei este prim.

Ucenicul vrăjitorului a cules  $N$  ciuperci dintre care unele sunt fermecate, altele nu. Vrăjitorul vrea să afle câte doze de licoare poate prepara din ciupercile culese, câte ciuperci fermecate îi rămân și câte ciuperci nu sunt bune de nimic. Scrieți un program care să-l ajute! Fiecare ciupercă are cel puțin două buline.

Exemplu:

$M=3$   $N=8$

$2$   
 $11$   
 $5$   
 $15$   
 $7$   
 $3$   
 $13$   
 $23$

doze: 2  
 ciuperci fermecate ramase: 1  
 ciuperci care nu sunt fermecate: 1

(Concurs "Grigore Moisil", Lugoj 2002, cls. V-VI)

20. (Câte sunt - \*\*\*\*\*) Fie mulțimea  $A=\{1, 2, \dots, n\}$ . Scrieți un program care să citească de la tastatură numărul natural  $n < 10.000$  și care afișează pe ecran câte numere raționale distincte de forma  $p/q$  cu  $p$  și  $q$  din  $A$  există.

Exemplu:

N=3		7
N=4		11
N=5		19
N=11		83

(Concurs "Grigore Moisil", Lugoj 2000, cls. VII-VIII)

21. (Factorial - \*) Fie  $N$  un număr natural ( $1.000 \leq N \leq 2.000.000.000$ ) despre care se știe că reprezintă factorialul unui număr  $k$  ( $N = 1*2*3*\dots*k$ ). Scrieți un program care să citească pe  $N$  și să determine numărul  $k$ .

Exemplu:

N=5040		k=7
N=3628800		k=10

22. (Suma minimă - \*) Se citește de la tastatură un număr natural  $n \leq 65.000$ . Numărul  $n$  se reprezintă în bazele de numerație de la 2 la 10. Să se determine bazele în care suma cifrelor reprezentării numărului  $n$  este minimă.

Exemplu:

n=13		2 3 6
------	--	-------

23. (Pitagora - \*\*\*\*) Se consideră  $a$  un număr natural nenul ( $a < 30000$ ). Să se găsească toate perechile de numere naturale  $b$  și  $c$  ( $a < b < c$ ) care împreună cu  $a$  formează triplete de numere pitagorice.

Exemplu:

a=9		12 15
		40 41

24. (Sfera - \*\*) Câte puncte cu coordonate întregi sunt conținute într-o sferă de rază  $R$  cu centrul în originea sistemului de coordonate? Se consideră ca  $R$  este un număr natural mai mic sau egal cu 30. Distanța  $d$  dintre un punct cu coordonatele  $(x, y, z)$  și originea sistemului de coordonate se determină după formula:

$$d = \sqrt{x^2 + y^2 + z^2}$$

Exemplu:

R=4		257
-----	--	-----

25. (Numere - \*\*\*) Scrieți un program care descompune numărul natural  $n < 51$  în sumă de numere naturale  $n = n_1 + n_2 + \dots + n_k$ , astfel încât produsul lor  $p = n_1 * n_2 * \dots * n_k$  să fie maxim.

Exemplu:

n=6		9
n=7		12
n=8		18

26. (Seiful - \*\*\*\*\*) Se consideră o comisie formată din  $n < 500$  persoane (numerate de la 1 la  $n$ ) care trebuie să păstreze într-un seif subiectele de la examenul de admitere. Să se determine numărul minim de lacăte necesar închiderii seifului astfel încât să existe o distribuție a cheilor lor care să îndeplinească următoarele condiții:

- oricare două persoane dețin același număr de chei
- fiecare persoană deține chei de la lacăte diferite
- toate lacătele seifului se vor putea deschide numai în prezența oricărui grup format din cel puțin  $n-1$  persoane

Pentru un lacăt pot exista mai multe chei care să-l deschidă. Nici o cheie nu poate deschide două lacăte diferite. Lacătele sunt numerotate de la 1 la  $l_{min}$ .

Să se determine numărul minim de lacăte, numărul total de chei distribuite și o repartizare a cheilor care respectă condițiile problemei.

Exemplu:

n=3		l <sub>min</sub> =3 chei=6
		lacat 1: 3 2
		lacat 2: 1 2
		lacat 3: 3 1

(Barajul lotului național de Informatică pentru IOI, 1998)

27. (Tort - \*\*\*\*) De ziua ei, Ionela a făcut un tort în formă triunghiulară. Dorind să-l împartă cu prietenii ei, Ionela efectuează  $A$  tăieturi dintr-un colț oarecare și  $B$  tăieturi din alt colț al tortului (un colț adică un vârf al triunghiului inițial). După multe astfel de tăieturi, ea este dezorientată: oare câte felii de tort în formă triunghiulară a tăiat? Prin felie înțelegem un triunghi cu tăieturi pe laturi și vârfuri în intersecțiile tăieturilor. O felie poate conține și alte tăieturi în interior.

Ajutați-o pe Ionela să numere feliile de tort în formă triunghiulară. Se știe că  $A, B$  sunt numere naturale mai mici ca 30.000.

Exemplu:

A=1 B=2		15
A=1 B=0		3

(<http://infoarena.devnet.ro>)

28. (Tabela - \*\*\*) Macarie, pasionat de numere și mai ales de matrice începe într-o zi să umple o foaie infinită de matematică (cu pătrățele) cu numere astfel: În colțul cel mai de sus stânga (1,1) pune 0, apoi scrie de la stânga la dreapta și de sus în jos cel mai mic număr care nu apare pe linia și coloana respectivă. Dându-se linia  $L$  și coloana  $C$  (numere naturale sub 2 miliarde) unei căsuțe din tabelă aflați numărul de la acea poziție.

0	1	2	3	4	5	...
1	0	3	2	5	4	...
2	3	0	1	6	7	...
3	2	1	0	7	6	...
...	...	...	...	...	...	...



Exemplu:

L=2 C=3  
L=4 C=5

3  
7

(http://infoarena.devnet.ro)

29. (Factorial - \*\*\*\*\*) Se dă un număr întreg  $P$  ( $0 < P < 10^8$ ). Problema cere găsirea celui mai mic număr natural strict pozitiv  $N$  pentru care produsul  $1*2*3*...*N$  are exact  $P$  cifre de 0 la sfârșit.

Exemplu:

P=2  
P=10

10  
45

(http://infoarena.devnet.ro)

30. (Rând - \*\*\*\*\*) Se consideră șirul infinit de numere: 3, 4, 5, 6, 7 8 ...

La fiecare mutare alegi primul număr din șir, fie acela  $x$ , și îl muți pe a  $x$ -a poziție.

3, 4, 5, 6, 7, 8 ... → 4, 5, 3, 6, 7, 8 ... → 5, 3, 6, 4, 7, 8 ... →

3, 6, 4, 7, 5, 8, ... → 6, 4, 3, 7, 5, 8 ... etc.

Să se determine pentru un număr natural  $n$  dat, care este cea mai mică mutare la care acesta apare pe prima poziție. Rezultatul va fi mai mic ca 2 miliarde.

Exemplu:

n=3  
n=4  
n=5  
n=6

1  
2  
3  
5

## CAPITOLUL 2

### Tipuri de Date Structurate

#### 2. 1. Tabloul unidimensional

##### 2.1.1 Teste cu alegere multiplă și duală

1. Identificați care din declarațiile următoare sunt corecte:

- |   |                                      |
|---|--------------------------------------|
| a) <code>a:array[1..100] of integer;</code>   | Ⓐ <code>int a[100];</code>           |
| b) <code>a:array 1..100 of byte;</code>       | Ⓑ <code>char a[1..100];</code>       |
| c) <code>a:array[1.00..10.00] of real;</code> | Ⓒ <code>float a[1.10..10.00];</code> |
| d) <code>a:array[1..100] of char;</code>      | Ⓓ <code>char a[100];</code>          |

2. Se considerăm următoarele declarații: `var a: array[1..5] of byte;` (varianta Pascal), respectiv `unsigned char a[4]` (varianta C/C++). Specificați valoarea elementelor tabloului după execuția instrucțiunii:

`for i:=1 to 5 do a[i]:=i-1;` | `for (i=0; i<5; i++) a[i]=i;`

- |              |              |             |            |
|--------------|--------------|-------------|------------|
| a) 1 1 1 1 1 | b) 0 0 0 0 0 | Ⓒ 0 1 2 3 4 | d) 1 2 3 4 |
|--------------|--------------|-------------|------------|

3. Se consideră următoarele declarații:

<code>var a:array[0..600] of real;</code>	<code>float a[600];</code>
<code>b:array[1..300] of char; i:byte;</code>	<code>char b[300]; unsigned char i;</code>

Care din următoarele referiri ale elementelor celor doi vectori sunt incorecte?

- |                        |                      |                       |                      |
|------------------------|----------------------|-----------------------|----------------------|
| a) <code>a[i*2]</code> | Ⓑ <code>a[-i]</code> | Ⓒ <code>b['l']</code> | d) <code>b[i]</code> |
|------------------------|----------------------|-----------------------|----------------------|

4. Fie declarația `var v: array[0..4] of integer;` (varianta Pascal), respectiv `int v[4];` (varianta C/C++). Specificați valoarea elementelor tabloului după execuția instrucțiunii:

`for i:=1 to 5 do v[i]:=2*(i-1);` | `for (i=0; i<5; i++) v[i]=2*i;`

- |               |             |              |              |
|---------------|-------------|--------------|--------------|
| a) 2 4 6 8 10 | Ⓑ 0 2 4 6 8 | c) 0 1 2 3 4 | d) 1 2 3 4 5 |
|---------------|-------------|--------------|--------------|

5. Se consideră următoarele declarații:

<code>var x:array[0..300] of char;</code>	<code>char x[301];</code>
<code>y:array[0..300] of byte; i:byte;</code>	<code>unsigned char y[301],i;</code>

Care din următoarele referiri ale elementelor celor doi vectori sunt incorecte?

- a) x[y[1]]    **b) y[(x[1])]L**    c) x['1']    d) x[300-y[0]]

6. Ce valori vor fi afișate în urma rulării următorului program?

```
var
  a:array[1..5] of integer;
  i:integer;
begin
  for i:=1 to 5 do a[i]:=i*10;
  for i:=2 to 5 do
    a[i]:=a[i]-a[i-1];
  for i:=1 to 5 do
    write(a[i], ' ')
end.
```

```
#include <iostream.h>
int a[5],i;
void main(){
  for (i=0;i<5;i++)
    a[i]=(i+1)*10;
  for (i=1;i<5;i++)
    a[i]=a[i]-a[i-1];
  for (i=0;i<5;i++)
    cout<<a[i]<<' ';
}
```

a) 10 20 30 40 40    b) 0 10 20 30 40    c) 10 10 10 10 10    **d) 10 10 20 20 30**

7. Ce valori vor fi afișate în urma rulării următorului program?

```
var a:array[1..6] of integer;
  i:integer;
begin
  for i:=1 to 6 do a[i]:=0;
  for i:=1 to 3 do a[2*i]:=i;
  for i:=1 to 6 do write(a[i], ' ')
end.
```

```
#include <iostream.h>
int a[6],i;
void main(){
  for (i=0;i<6;i++) a[i]=0;
  for (i=0;i<3;i++) a[2*i+1]=i+1;
  for (i=0;i<6;i++)
    cout<<a[i]<<' ';
}
```

**a) 0 1 0 2 0 3**    b) 1 0 2 0 3 0    c) 0 2 0 4 0 6    d) 1 2 3 4 5 6

8. Ce valori vor fi afișate în urma rulării următorului program?

```
var a:array[1..5] of integer;
  i:integer;
begin
  for i:=1 to 5 do a[i]:=i-1;
  write(a[2], ' ');
  write(a[a[2]], ' ');
  write(a[a[a[3]]], ' ');
end.
```

```
#include <iostream.h>
int a[6],i;
void main(){
  for (i=1;i<=5;i++) a[i]=i-1;
  cout<<a[2]<<' ';
  cout<<a[a[2]]<<' ';
  cout<<a[a[a[3]]]<<' ';
}
```

a) 1 0 1    b) 1 1 1    c) 1 2 0    **d) 1 0 0**

9. Ce valori vor fi afișate în urma rulării următorului program?

```
var a,b:array[1..5] of integer;
  i:integer;
```

```
#include <iostream.h>
int a[5],b[5],i;
```

```
begin
  for i:=1 to 5 do
    if i mod 2<>0 then a[i]:=1
    else a[i]:=i+1;
  for i:=1 to 5 do b[i]:=a[i]*2;
  for i:=1 to 5 do write(b[i], ' ')
end.
```

```
void main(){
  for (i=0;i<5;i++)
    if (i%2==0) a[i]=1;
    else a[i]=i+2;
  for (i=0;i<5;i++) b[i]=a[i]*2;
  for (i=0;i<5;i++)
    cout<<b[i]<<' ';
}
```

- a) 1 6 1 10 1    **b) 2 6 2 10 2**    c) 2 2 6 2 2    d) 2 4 6 8 10

10. Care dintre următoarele secvențe de instrucțiuni determină în mod corect elementul maxim din vectorul *a: array[1..5] of integer* (varianta Pascal), respectiv *int a[4];* (varianta C/C++).?

- a) max:=a[1];  
for i:=2 to 5 do  
if a[i]>max then max:=a[i];
- b) max:=0;  
for i:=1 to 5 do  
if a[i]>max then max:=a[i];
- c) max:=a[n];  
for i:=n downto 1 do  
if a[i]>max then max:=a[i];
- d) max:=30000;  
for i:=5 downto 1 do  
if a[i]>max then max:=a[i];
- a) max:=a[0];  
for (i=1;i<5;i++)  
if (a[i]>max) max=a[i];**
- b) max:=0;  
for (i=0;i<5;i++)  
if (a[i]>max) max=a[i];
- c) max:=a[n-1];  
for (i=n-1;i>=0;i--)  
if (a[i]>max) max=a[i];**
- d) max:=30000;  
for (i=4;i>=0;i--)  
if (a[i]>max) max=a[i];

11. Care dintre următoarele instrucțiuni realizează deplasarea cu o poziție spre dreapta a tuturor elementelor tabloului *a* începând cu cel situat pe poziția *p*. Valoarea variabilei întregi *n* reprezintă indicele ultimului element din vectorul *a*.

- a) for i:=p to n do  
a[i+1]:=a[i];
- b) for i:=p+1 to n+1 do  
a[i]:=a[i-1];
- c) for i:=n+1 to downto p do  
a[i]:=a[i-1];
- d) for i:=n downto p do  
a[i+1]:=a[i];
- a) for(i=p;i<=n;i++)  
a[i+1]=a[i];**
- b) for(i=p+1;i<=n+1;i++)  
a[i]=a[i-1];
- c) for(i=n+1;i>=p;i--)  
a[i]=a[i-1];**
- d) for(i=n;i>=p;i--)  
a[i+1]=a[i];**

12. Care dintre următoarele instrucțiuni realizează deplasarea cu o poziție spre stânga a tuturor elementelor tabloului *a* începând cu cel situat pe poziția *p+1*, în vederea ștergerii elementului de pe poziția *p*? Valoarea variabilei întregi *n* reprezintă indicele ultimului element din *a*.

- a) for i:=p+1 to n do  
a[i]:=a[i-1];
- b) for i:=p to n-1 do  
a[i]:=a[i+1];
- c) for i:=n to downto p+1 do  
a[i-1]:=a[i];
- d) for i:=n downto p do  
a[i+1]:=a[i];
- a) for(i=p+1;i<=n;i++)  
a[i]=a[i-1];**
- b) for(i=p;i<=n;i++)  
a[i]=a[i+1];**
- c) for(i=n;i>=p;i--)  
a[i-1]=a[i];
- d) for(i=n;i>=p;i--)  
a[i+1]=a[i];

13. Considerând următoarele instrucțiuni identificați două echivalente:

- |   |  |
|---|--|
| a) for i:=1 to n do<br>if odd(i) then s:=s+a[i];          | a) for(i=0;i<n;i++)<br>if (i%2!=0) s=s+a[i];     |
| b) for i:=1 to n div 2 do<br>s:=s+ a[i*2-1];              | b) for(i=1;i<=n/2;i++)<br>s=s+a[i*2-1];          |
| c) for i:=n to downto 2 do<br>if odd(i) then s:=s+a[i-1]; | c) for(i=n-1;i>1;i--)<br>if (i%2!=0) s=s+a[i-1]; |
| d) for i:=2 to (n+1)div 2 do<br>s:=s+a[i*2-3];            | d) for(i=2;i<=(n+1)/2;i++)<br>s=s+a[i*2-3];      |

14. Care dintre declarațiile de mai jos reprezintă declararea corectă a unei variabile de tip tablou unidimensional cu 20 elemente întregi cu semn:

- |  |                        |
|--|------------------------|
| a) type<br>as=array[1..20] of integer;<br>var a: as; | a) int a[20];          |
| b) var<br>a=array[1..20] of integer;                 | b) int a[ ];           |
| c) var<br>a:array[1..20]of -33760...33767;           | c) unsigned int *a;    |
| d) var<br>a:array[1..20] of byte;                    | d) unsigned int a[20]; |
| e) var<br>a:array['a'..'t'] of integer;              | e) short int a[20];    |

15. Fie  $a$  un vector cu  $n=9$  componente întregi. Ce va afișa pentru  $a=(14, 3, 7, 0, -4, 3, 10, 15, 7)$ ?

- |   |  |
|---|--|
| s:=0;<br>for i:=1 to n do<br>if odd(i) then s:=s+a[i];<br>writeln(s); | s=0;<br>for(i=0;i<n;i++)<br>if (i%2==0) s+=a[i];<br>cout<<s; |
| a) 0  | b) 34  |
| c) 21   | d) 50  |

### 2.1.2 Teste cu itemi semiobiectivi

1. Se consideră următorul program pseudocod:

```

1  întreg i,n,a[100];
2  citește n;
3  pentru i←1, n executa
4  │ citește a[i];
5  │
6  │ pentru i←1, n-1 executa
7  │ │ dacă a[i]>a[i+1] atunci
8  │ │ │ a[i] ↔ a[i+1];
9  │ │
10 │
```

- a) Ce se va afișa pentru  $n=6$  și tabloul  $a=(2, 0, 1, 4, 6, 3)$ ?
- b) Dați un exemplu de set de date de intrare pentru care nu se va efectua nici o interschimbare.

```

11 │ pentru i←1, n executa
12 │ │ scrie a[i];
13 │ │ stop.
14 │
15 │
```

c) Dați exemplu de set de date de intrare cu valori neordonate, pentru care se va afișa un șir de valori ordonate crescător.

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

2. Se consideră următorul program pseudocod:

```

1  întreg i, n, a[100];
2  citește n;
3  pentru i←1, n executa
4  │ citește a[i];
5  │
6  │ pentru i←2, n-1 executa
7  │ │ dacă a[i] div 10=0 atunci
8  │ │ │ a[i] ← a[i-1] + a[i+1];
9  │ │ │
10 │ │
11 │ │ pentru i←1, n executa
12 │ │ │ scrie a[i];
13 │ │ │ stop.
14 │ │
15 │ │
16 │ │
17 │ │
18 │
```

a) Ce se va afișa pentru  $n=5$  și tabloul  $a=(22, 4, 10, 5, 16)$ ?

b) Dați un exemplu de set de date de intrare pentru care nu se va efectua nici o modificare a valorilor tabloului.

c) Rescrieți instrucțiunea dacă atunci folosind o condiție (expresie logică) echivalentă. Realizați două astfel de instrucțiuni echivalente cu cea prezentată.

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

3. Se consideră următorul program pseudocod:

```

1  întreg i, n, j, k, a[100];
2  citește n;
3  pentru i←1, n executa
4  │ citește a[i];
5  │
6  │ i ← 1; j ← n;
7  │ cat timp(a[i]=a[j]) and (i<j)
8  │ │ │ execută
9  │ │ │ │ i ← i + 1; j ← j - 1;
10 │ │ │
11 │ │ pentru k ← i, j execută
12 │ │ │ scrie a[k];
13 │ │ │ stop.

```

a) Ce se va afișa pentru  $n=8$  și tabloul  $a=(2, 2, 3, 4, 1, 2, 2, 2)$ ?

b) Dați un exemplu de set de date de intrare pentru care se va afișa același tablou cu cel citit.

c) Dați exemplu de set de date de intrare pentru care nu se va afișa nici o valoare.

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

4. Se consideră următorul program pseudocod:

```

1  intreg i, n, x, a[100];
2  citește x;
3  n ← 0;
4  cat timp x ≠ 0 executa
5      n ← n + 1;
6      a[n] ← x mod 10;
7      x ← x div 10;
8  ■
9  pentru i ← 1, n executa
10     scrie a[i];
11  ■ stop.
12
13

```

a) Ce se va afișa pentru  $x=20341$ ?

b) Dați un exemplu de dată de intrare pentru care se va afișa un șir de cifre identic cu cel citit.

c) Rescrieți algoritmul folosind structura repetitivă condiționată posterior, în locul structurii *cat timp*.

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

5. Se consideră următorul program pseudocod:

```

1  intreg i, n, nr, a[100];
2  citește n; nr ← 0;
3  pentru i ← 1, n executa
4      citește a[i];
5  ■
6  pentru i ← 1, n-1 executa
7      dacă a[i]*a[i+1] < 0 atunci
8          nr ← nr + 1;
9  ■
10 ■
11 dacă nr = 0 atunci scrie 'DA'
12     altfel scrie 'NU'
13 ■ stop.
14

```

a) Ce se va afișa pentru  $n=6$  și tabloul  $a=(3, -1, -4, 2, 4, 5)$ ?

b) Dați un exemplu de set de date de intrare pentru care variabila *nr* va primi valoarea  $n-1$ .

c) Dați exemplu de set de date de intrare pentru care se va afișa mesajul 'DA'.

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

6. Se consideră următorul program pseudocod:

```

1  citește n;
2  pentru i ← 1, n executa
3      citește a[i];
4  ■
5  i ← 2;
6  cat timp i < n executa
7      dacă a[i]=a[i-1]+a[i+1] atunci
8          pentru j ← i, n executa
9              a[j] ← a[j+1];
10         ■
11         n ← n - 1;
12         altfel i ← i + 1;
13     ■
14 ■

```

a) Ce valori vor fi afișate pentru  $n=5$  și vectorul  $a=(2,5,3,6,1)$ ? Dar pentru  $n=5$  și vectorul  $a=(2,5,3,1,-1)$ ?

b) Dați un exemplu pentru datele de intrare astfel încât să se afișeze valorile din vector nemodificate.

c) Realizați un enunț de problemă a cărei rezolvare este algoritmul prezentat.

```

15  pentru i ← 1, n executa
16     scrie a[i];
17  ■
18

```

d) Pentru  $n>1$ , există un set de valori pentru elementele vectorului *a*, astfel încât în final algoritmul să afișeze o singură valoare?

e) Realizați programul în limbajul de programare studiat Pascal/C/C++.

7. Se consideră următorul program pseudocod:

```

1  intreg nr[10], c, max, i, n, j, x;
2  citește n; max ← 0;
3  pentru i ← 0, 9 executa
4      nr[i] ← 0;
5  ■
6  pentru i ← 1, n executa
7      citește j;
8      cat timp j ≠ 0 executa
9          x ← j mod 10;
10         nr[x] ← nr[x] + 1;
11         dacă max < nr[x] atunci
12             max ← nr[x]; c ← x;
13         ■
14         j ← j div 10;
15     ■
16 ■
17 scrie c;

```

a) Ce valori vor fi afișate pentru  $n=5$  și valorile 22, 235, 233, 6, 221?

b) Dați un exemplu pentru datele de intrare (nu toate nule), astfel încât să se afișeze valoarea 0

c) Realizați un enunț de problemă a cărei rezolvare este algoritmul prezentat.

d) Rescrieți algoritmul folosind structura condiționată posterior în locul structurii *cat timp*.

e) Realizați programul în limbajul de programare studiat Pascal/C/C++.

8. Se consideră următorul program pseudocod:

```

1  intreg a[100], max, i, n;
2  citește n; max ← 1;
3  pentru i ← 1, n executa
4      citește a[i];
5  ■
6  scrie a[1];
7  pentru i ← 2, n executa
8      dacă a[i] < a[i-1] atunci
9          max ← max + 1;
10         scrie '*', a[i];
11     altfel
12         scrie a[i];
13     ■
14 ■
15 scrie max;

```

a) Ce valori vor fi afișate pentru  $n=7$  și valorile 2, 2, 34, 5, 6, 78, 8?

b) Dați un exemplu pentru datele de intrare (nu toate nule), astfel încât să nu se afișeze nici un caracter '\*'.

c) Realizați un enunț de problemă a cărei rezolvare este algoritmul prezentat.

d) Care este numărul maxim de caractere '\*' care pot fi afișate pentru un tablou cu *n* elemente?

e) Realizați programul în limbajul de programare studiat Pascal/C/C++.

9. Se consideră următorul program pseudocod:

```

1  integ A[100], max, i, n;
2  citește n;
3  p ← 0;
4  pentru i ← 1, n executa
5  | citește a[i];
6  |
7  i ← 1;
8  j ← n;
9  cat timp (i≠j) and (p=0) executa
10 | m ← (i+j) div 2;
11 | dacă a[m] = x atunci p ← m
12 | altfel
13 |   dacă a[m] > x atunci
14 |     j ← m - 1
15 |   altfel
16 |     i ← m + 1;
17 |
18 |
19 |
20 scrie p;

```

a) Ce valori vor fi afișate pentru  $n=7$ ,  $x=5$  și valorile 3, 5, 6, 7, 8, 9, 23? Dar pentru  $n=7$ ,  $x=15$  și valorile 3, 5, 6, 7, 8, 9, 23?

b) Ce semnificație dați valorii variabilei  $m$ ? Dar variabilei  $p$ ?

c) Realizați un enunț de problemă a cărei rezolvare este algoritmul prezentat.

d) Câte operații de comparare se efectuează maxim în cazul unui vector de cu 1024 de valori ordonate crescător?

e) Realizați programul în limbajul de programare studiat Pascal/C/C++.

10. Se consideră următorul program pseudocod:

```

1  integ a[100], max, i, n;
2  citește n, a[1];
3  nr ← 1;
4
5  pentru i ← 2, n executa
6  | citește x;
7  | k ← 1;
8  | cat timp (x ≤ a[k]) and (k ≤ nr)
9  |   executa
10 |   k ← k + 1;
11 |
12 |
13 | dacă k = nr + 1 atunci
14 |   nr ← nr + 1;
15 |
16 | a[k] ← x;
17 |
18 |
19 pentru i ← 1, nr executa
20 | scrie a[i];
21 |
22 |

```

a) Ce valori vor fi afișate pentru  $n=7$  și valorile 3, 5, 6, 4, 8, 9, 2? Dar pentru  $n=7$  și valorile 3, 5, 6, 7, 8, 9, 23?

b) Dați exemplu de valori pentru care se va afișa șirul de valori 8 5 3 12.

c) Ce semnificație dați valorii variabilei  $nr$ ?

d) Considerând o valoare oarecare pentru  $n$ , cum trebuie să fie șirul valorilor introduse pentru ca la final să se afișeze tot cele  $n$  valori?

e) Realizați programul în limbajul de programare studiat Pascal/C/C++.

### 2.1.3 Probleme rezolvate

1. Fie un tablou unidimensional care conține  $n$  valori întregi. Realizați un program care ordonează crescător elementele vectorului folosind "algoritmul de selecție".

**Soluție:** În cazul sortării prin selecție, elementele vectorului sunt divizate în două "liste", una ordonată, iar cealaltă neordonată. Considerăm că ele sunt despărțite de un "perete" imaginar.

Algoritmul de selecție se bazează pe  $n-1$  pași succesivi de tipul:

- identificarea celui mai mic element al listei neordonate
- interschimbarea minimului cu primul element al listei neordonate.

După această selecție (identificare a elementului minim) și interschimbare, "peretele" despărțitor se va deplasa cu un element înainte. În felul acesta se mărește cu 1 numărul de elemente al listei ordonate și se micșorează cu 1 numărul de elemente al listei neordonate.

De câte ori se efectuează această deplasare, spunem că s-a încheiat un pas complet al sortării. Dacă avem  $n$  elemente în vector, vor exista  $n-1$  astfel de pași.

<pre> 1  var 2  a:array[1..100] of integer; 3  i,n,p,j,aux:integer; 4  begin 5  readln(n); 6  for i:=1 to n do read(a[i]); 7  for i:=1 to n-1 do begin 8    p:=i; 9    for j:=i+1 to n do 10     if a[p]&gt;a[j] then p:=j; 11     aux:=a[i]; 12     a[i]:=a[p]; 13     a[p]:=aux; 14   end; 15 for i:=1 to n do write(a[i], ' '); 16 end. </pre>	<pre> #include &lt;iostream.h&gt; int a[100],i,n,p,j,aux; void main() {   cin&gt;&gt;n;   for (i=0;i&lt;n;i++) cin&gt;&gt;a[i];   for (i=0;i&lt;n;i++) {     p=i;     for (j=i+1;j&lt;n;j++)       if (a[p]&gt;a[j]) p=j;     aux=a[i]; a[i]=a[p];     a[p]=aux;   }   for (i=0;i&lt;n;i++)     cout&lt;&lt;a[i]&lt;&lt;' '; } </pre>
---	---

2. Fie un tablou unidimensional care conține  $n$  valori întregi. Realizați un program care ordonează crescător elementele vectorului folosind "algoritmul de sortare cu bule" - (Bubble Sort).

**Soluție:** În cadrul acestui algoritm, elementele vectorului sunt împărțite în două liste: sortată și nesortată. Cel mai mic element este "ridicat" din lista nesortată și mutat în cea ordonată.

După această mutare, "peretele" care desparte cele două liste înaintea cu o poziție, în acest fel mărimu-se cu un element lista sortată și micșorându-se cu un element cea nesortată. La fiecare pas de acest tip se consideră că s-a încheiat un pas complet al sortării.

Considerând că vectorul are  $n$  elemente atunci sortarea necesită  $n-1$  astfel de pași.

```
1 var a:array[1..100]of integer; #include <iostream.h>
2 i,n,j,aux:integer; int a[100],i,n,j,aux;
3 begin void main() {
4 readln(n); cin>>n;
5 for i:=1 to n do read(a[i]); for (i=0;i<n;i++) cin>>a[i];
6 for i:=1 to n do for (i=0;i<n;i++)
7 for j:=n downto i+1 do for (j=n-1;j>i;j--)
8 if a[j]<a[j-1] then begin if (a[j]<a[j-1]) {
9 aux:=a[j]; a[j]:=a[j-1]; aux=a[j]; a[j]=a[j-1];
10 a[j-1]:=aux; a[j-1]=aux;
11 end; }
12 for i:=1 to n do write(a[i], ' '); for (i=0;i<n;i++)
13 end. cout<<a[i]<<' ';
14 }
```

3. Fie un tablou unidimensional care conține  $n$  valori întregi. Realizați un program care ordonează crescător elementele vectorului folosind "algoritmul de inserție" - (Insertion Sort).

**Soluție:** Algoritmul de sortare prin inserție funcționează pe aceeași idee de a divide vectorul în două liste: ordonată și neordonată. La fiecare pas, primul element al listei nesortate este transferat în cea sortată, exact pe poziția prin care se respectă ordinea crescătoare a elementelor. Această operație se va efectua prin deplasarea cu o poziție spre dreapta a tuturor elementelor mai mari decât el.

```
1 var a:array[1..100]of integer; #include <iostream.h>
2 i,n,j,aux:integer; ok:boolean; int a[100],i,n,j,aux,ok;
3 begin void main() {
4 readln(n); cin>>n;
5 for i:=1 to n do read(a[i]); for (i=0;i<n;i++) cin>>a[i];
6 for i:=2 to n do begin for (i=1;i<n;i++) {
7 aux:=a[i]; j:=i-1; aux=a[i]; j:=i-1; ok:=0;
8 ok:=false; while (j>=0 && !ok) {
9 while (j>=1)and(not ok) do if (aux<a[j]) {
10 if aux<a[j] then begin a[j+1]=a[j]; j:=j-1;
11 a[j+1]:=a[j]; j:=j-1; }
12 end; else ok=1;
13 else ok:=true; a[j+1]=aux;
14 a[j+1]:=aux; for (i=0;i<n;i++)
15 end; cout<<a[i]<<' ';
16 for i:=1 to n do write(a[i], ' ');
17 end. }
```

4. Fie un tablou unidimensional care conține  $n$  valori întregi distincte. Realizați un program care ordonează crescător elementele vectorului folosind "algoritmul de numărare".

**Soluție:** Considerăm vectorul  $A$ . Algoritmul de sortare prin numărare constă în găsirea pentru fiecare element  $a[i]$ , a numărului de elemente din vector mai mici ca el. Numerele obținute sunt memorate într-un alt vector  $B$ . Elementele vectorului  $A$  vor fi inițial salvate în vectorul auxiliar  $C$ . La finalul algoritmului se vor rescrie în ordine crescătoare elementele vectorului  $A$  pe baza valorilor memorate în  $B$  și  $C$ .

```
1 var #include <iostream.h>
2 a,b,c:array[1..100]of integer; int a[100],b[100],c[100];
3 i,n,j,aux:integer; ok:boolean; int i,n,j,aux,ok;
4 begin void main() {
5 readln(n); cin>>n;
6 for i:=1 to n do read(a[i]); for (i=0;i<n;i++) cin>>a[i];
7 for i:=1 to n do begin for (i=0;i<n;i++) {
8 b[i]:=0; c[i]:=a[i]; end; b[i]=0; c[i]=a[i];
9 for i:=2 to n do }
10 for j:=1 to i-1 do for (i=1;i<n;i++)
11 if c[j]<c[i] then inc(b[i]); for (j=0;j<i;j++)
12 else inc(b[j]); if (c[j]<c[i]) b[i]++;
13 for i:=1 to n do else b[j]++;
14 a[1+b[i]]:= c[i]; for (i=0;i<n;i++)
15 for i:=1 to n do,write(a[i], ' '); a[b[i]]=c[i];
16 end. for (i=0;i<n;i++)
17 cout<<a[i]<<' '; }
```

5. Fie  $N$  un număr natural de cel mult 9 cifre. Să se genereze toate numerele care se pot scrie ca produs de două numere prime mai mici ca  $N$ .

**Soluție:** Pentru optimizarea algoritmului de generare a numerelor prime vom proceda astfel: variabila  $p$  va parcurge toate numerele impare mai mici ca  $n$ . Pentru a verifica dacă  $p$  este un număr prim se va testa dacă unul din numerele prime mai mici ca el reprezintă un divizor al lui. Vectorul  $prim$  va reține toate numerele prime mai mici decât  $N$ . Valorile care se vor afișa în final se vor obține ca produse a două elemente ale vectorului  $prim$ .

```
1 var prim:array[1..100]of #include <iostream.h>
2 integer; m,i,p,n,j:integer; int prim[100],m,n,i,p,j;
3 begin void main() {
4 readln(n); cin>>n;
5 prim[1]:= 2; m:= 1; p:= 3; prim[0]=2;m=1;p=3;
6 while p<=n do begin while (p<=n) {
7 i:=1; i=0;
8 while i<=m do while (i<m)
9 if p mod prim[i]=0 then i:=n; if (p%prim[i]==0) i:=n;
10 else inc(i); else
11 if i<n then begin i++;
12 inc(m); prim[m]:=p; end; if (i!=n) prim[m++]=p;
13 p :=p + 2 ; p+=2;
14 end; }
```

```

15 for i:=1 to m do
16   for j:=i to m do
17     writeln(prim[i]*prim[j])
18 end.

```

```

for (i=0;i<m;i++)
  for (j=i;j<m;j++)
    cout<<prim[i]*prim[j]<<endl;
}

```

6. Se consideră  $n$  intervale închise  $[a, b]$ . Să se determine reuniunea acestora.

**Soluție:** Vom reține fiecare capăt stâng al intervalelor în vectorul  $A$  iar cel drept în vectorul  $B$ . Primul pas al algoritmului va ordona intervalele în funcție de capătul stâng al acestora. Pentru determinarea reuniunii, este suficientă o parcurgere linjară a acestora, păstrând la fiecare moment capătul stâng și drept al intervalului curent al reuniunii, actualizând la nevoie valorile acestora.

```

1 var a,b:array[1..100] of integer;
2 i,n,j,x,y:integer;
3 begin
4   readln(n);
5   for i:=1 to n do read(a[i],b[i]);
6   for i:=1 to n-1 do
7     for j:=i+1 to n do
8       if a[i]>a[j] then begin
9         x:=a[i];a[i]:=a[j];a[j]:=x;
10        x:=b[i];b[i]:=b[j];b[j]:=x;
11      end;
12   x:=a[1];y:=b[1];
13   for i:=2 to n do
14     if a[i]>y then begin
15       writeln(x,' ',y);
16       x:=a[i];y:=b[i]; end
17     else
18       if b[i]>y then y:=b[i];
19   writeln(x,' ',y)
20 end.

```

```

#include <iostream.h>
int a[100],b[100],i,n,j,x,y;
void main() {
  cin>>n;
  for (i=0;i<n;i++)
    cin>>a[i]>>b[i];
  for (i=0;i+1<n;i++)
    for (j=i+1;j<n;j++)
      if (a[i]>a[j]) {
        x=a[i]; a[i]=a[j]; a[j]=x;
        x=b[i]; b[i]=b[j]; b[j]=x;
      }
  x=a[0]; y=b[0];
  for (i=1;i<n;i++)
    if (a[i]>y) {
      cout<<x<<' '<<y<<endl;
      x=a[i]; y=b[i];
    }
    else if (b[i]>y) y=b[i];
  cout<<x<<' '<<y<<endl;}

```

7. Se consideră un tablou unidimensional ce conține  $n$  caractere distincte. Să se afișeze permutarea circulară a lui, care începe cu cel mai mic caracter în sens lexicografic.

**Exemplu:** Pentru  $n=5$  și tabloul ('m','i','o','a','r') se va afișa ('a','r','m','i','o').

**Soluție:** Se va determina poziția elementului minim în vector, după care toate elementele din fața lui se vor copia la final, începând cu poziția  $n+1$ . Ștergerea elementelor din fața minimului se va face prin deplasări spre stânga ale celorlalte prezente în vector.

```

1 var a:array[1..100] of char;
2 i,j,n,p:integer;
3 min:char;
4 begin
5   readln(n);readln(a[1]);
6   p:=1; min:=a[1];

```

```

#include <iostream.h>
char a[100],min;
int i,j,n,p;
void main() {
  cin>>n; cin>>a[0];
  p=0; min=a[0];

```

```

7 for i:=2 to n do begin
8   readln(a[i]);
9   if a[i]<min then begin
10     min:=a[i];p:=i;
11   end;
12 end;
13 for i:=1 to p-1 do a[n+i]:=a[i];
14 for i:=p to n+p-1 do
15   a[i-p+1]:=a[i];
16 for i:=1 to n do write(a[i])
17 end.

```

```

for (i=1;i<n;i++) {
  cin>>a[i];
  if (a[i]<min) {
    min=a[i]; p=i;
  }
  for (i=0;i<p;i++) a[n+i]=a[i];
  for (i=p;i<n+p;i++)
    a[i-p]=a[i];
  for (i=0;i<n;i++) cout<<a[i];
}

```

8. Fie o mulțime ce conține  $n$  elemente ( $n < 25$ ). Să se afișeze toate submulțimile acestei mulțimi.

**Soluție:** Să considerăm mulțimea reprezentată cu ajutorul vectorului  $A=(a[1],a[2],\dots,a[n])$ .

Notăm cu  $B$  o submulțime a mulțimii  $A$ . Vectorului caracteristic al submulțimii cuprinde  $n$  elemente având valori din mulțimea  $\{0,1\}$ .

$$c[i] = \begin{cases} 1 & \text{daca } a[i] \in A \\ 0 & \text{daca } a[i] \notin A \end{cases}$$

Considerăm mulțimea reprezentată cu ajutorul vectorului  $(3,1,4,5,2,8)$ . Submulțimea  $\{2,4,1\}$  poate fi reprezentată cu ajutorul vectorului caracteristic  $(0,1,1,0,1,0)$ .

Această operație este posibilă prin simularea adunării în baza 2 dintre 1 și numărul ale cărui cifre sunt elementele vectorului caracteristic. Reamintim că  $1+1=10_2$  și  $1+0=1_2$ .

- 1) Se începe cu vectorul caracteristic al submulțimii vide.
- 2) Pornind de la elementul  $c[n]$ , atâta timp cât se întâlnesc elemente egale cu 1 ele se transformă în zero ( $1+1=10_2$ ). Primul element egal cu 0 se transformă în 1.
- 3) Se tipăresc elementele submulțimii reprezentate de vectorul caracteristic obținut, după care se revine la pasul 2.

Algoritmul se termină când a fost generat vectorul caracteristic cu toate elementele egale cu 1.

```

1 var c,a:array[1..100] of byte;
2 n,i:integer;
3 ok:boolean;
4
5 begin
6   read(n);
7   for i:=1 to n do read(a[i]);
8   fillchar(c,n,0);
9   ok:=true;

```

```

#include <iostream.h>
unsigned char a[100],c[100];
int n,i,ok;

void main() {
  cin>>n;
  for (i=0;i<n;i++) cin>>a[i];
  for (i=0;i<n;i++) c[i]=0;
  ok=1;

```

```

10 while ok do begin
11   i:=n;
12   while (c[i]=1)and(i>0) do
13     begin
14       c[i]:=0;
15       i:=i-1;
16     end;
17   if i=0 then ok:=false
18   else begin
19     c[i]:=1;
20     writeln;
21     for i:=1 to n do
22       if c[i]=1 then write(a[i], ' ')
23     end;
24   end;
25 end.

```

```

while (ok) (
  i:=n-1;
  while (c[i]=1 && i>0) {
    c[i]:=0;
    i--;
  }
  if (i=-1) ok=0;
  else {
    c[i]=1;
    cout<<endl;
    for (i=0;i<n;i++)
      if (c[i]) cout<<a[i]<<' ';
  }
}
)

```

9. Se consideră un vector cu  $n$  elemente întregi. Să se elimine cât mai puține elemente de la extremitățile vectorului astfel încât cel două valori rămase la "capete" să fie consecutive.

*Exemplu:* Pentru  $n=9$  și vectorul (8,2,4,5,2,5,3,4,6) se va afișa: (2,4,5,2,5,3) deoarece s-a eliminat primul element și ultimele două. Valorile rămase la capete sunt consecutive.

*Soluție:* Algoritmul determină pentru fiecare element din vector poziția elementului "pereche", adică a elementului cel mai apropiat de ultimul element, cu proprietatea ca valoarea absolută a diferenței lor este egală cu 1.

```

1 var a:array[1..100]of byte;
2   min,n,i,j,p1,p2,x:integer;
3 begin
4   read(n); min:=n; p1:=1; p2:=0;
5   for i:=1 to n do read(a[i]);
6   for i:=1 to n do begin
7     for j:=n downto i do
8       if abs(a[i]-a[j])=1 then
9         begin
10          x:=j; break;
11        end;
12   if n-x<min then begin
13     min:=n-x; p1:=i; p2:=x;
14   end;
15 end;
16 for i:=p1 to p2 do
17   write(a[i], ' ')
18 end.
19
#include <iostream.h>
#include <math.h>
unsigned char a[100];
int min,n,i,j,p1,p2,x;
void main() {
  cin>>n;
  min=n; p1=0; p2=-1;
  for (i=0;i<n;i++) cin>>a[i];
  for (i=0;i<n;i++) {
    for (j=n-1;j>=i;j--)
      if (abs(a[i]-a[j])=1) {
        x=j; break;
      }
    if (n-x<min)
      ( min=n-x; p1=i; p2=x; )
  }
  for (i=p1;i<=p2;i++)
    cout<<a[i]<<' ';
}

```

10. Se consideră doi vectori  $A$  și  $B$  cu  $n$  elemente valori naturale. Se știe că toate elementele lor se pot grupa în  $n$  perechi de forma  $(a[i], b[j])$ , astfel încât suma  $a[i]+b[j]$  să fie aceeași pentru orice pereche.

Scrieți un program care să determine suma și elementele fiecărei perechi. De la tastatură se citesc: numărul  $n$  și elementele celor doi vectori.

*Exemplu:* Pentru  $n=4$ ,  $A=4\ 2\ 1\ 5$  și respectiv  $B=(5\ 3\ 2\ 6)$  se va afișa:

```

7
43
25
16
52

```

*Soluție:* Suma unei perechi se obține prin însumarea tuturor elementelor vectorilor  $A$  și  $B$  și împărțirea acestei valori la  $n$ . O pereche va fi formată din elementul  $a[i]$  și  $S-a[i]$  (care se regăsește ca element în vectorul  $B$ , conform enunțului).

```

1 var j,i,n,s:integer;
2   a,b:array[1..100]of integer;
3 begin
4   s:=0; readln(n);
5   for i:=1 to n do begin
6     readln(a[i]);
7     s:=s+a[i];
8   end;
9   for i:=1 to n do begin
10    readln(b[i]); s:=s+b[i];
11  end;
12  s:=s div n;
13  writeln(s);
14  for i:=1 to n do
15    writeln(a[i], ' ', s-a[i]);
16 end.

#include <iostream.h>
int j,i,n,a[100],b[100],s=0;
void main() {
  cin>>n;
  for (i=0;i<n;i++) {
    cin>>a[i]; s+=a[i];
  }
  for (i=0;i<n;i++) {
    cin>>b[i]; s+=b[i];
  }
  s/=n;
  cout<<s<<endl;
  for (i=0;i<n;i++)
    cout<<a[i]<<" "<<
      s-a[i]<<endl;
}

```

11. Afișați cifrele distincte ale unui număr în ordine crescătoare a numărului lor de apariții. *Exemplu:* Pentru  $n=21223$  se va afișa 1 3 2

*Soluție:* Pentru codificarea datelor se vor folosi doua tablouri unidimensionale cu indici între 0 și 9:

- vectorul  $a$  în care elementul  $a[i]$  va indica numărul de apariții ale cifrei  $i$  în scrierea zecimală a numărului  $x$ , citit ca dată de intrare;
- vectorul  $c$  în care elementul  $c[i]$  face asocierea între cifra  $i$  și poziția  $i$  prin atribuirea  $c[i] \leftarrow i$ .

La ordonarea vectorului  $a$  se vor interschimba și elementele corespunzătoare din vectorul  $c$ .

În final, afișarea elementelor vectorului  $c$  se va face numai pentru cifrele al căror număr de apariție este strict mai mare decât 0.

```

1 var p,j,i,x:longint;
2   a,c:array[0..9]of byte;
3 begin
4   readln(x);
5   for i=0 to 9 do a[i]:=0;

#include <iostream.h>
long p,j,i,x,a[10],c[10];
void main() {
  cin>>x;
  for (i=0;i<=9;i++) a[i]=0;
}

```



```

6 for i=0 to 9 do c[i]:=i;
7 repeat
8   inc(a[x mod 10]);
9   x:=x div 10;
10 until x=0;
11 for i:=0 to 8 do
12   for j:=i+1 to 9 do
13     if (a[i]>a[j]) then begin
14       p:=a[i];a[i]:=a[j];a[j]:=p;
15       p:=c[i];c[i]:=c[j];c[j]:=p;
16     end;
17   for i:=0 to 9 do
18     if a[i]>0 then write(c[i]);
19 end.

```

```

for (i=0;i<=9;i++) c[i]=i;
do {
  a[x%10]++;
  x/=10;
} while (x!=0);
for (i=0;i<=8;i++)
  for (j=i+1;j<=9;j++)
    if (a[i]>a[j]) {
      p=a[i];a[i]=a[j];a[j]=p;
      p=c[i];c[i]=c[j];c[j]=p;
    }
for (i=0;i<=9;i++)
  if (a[i]>0) cout<<c[i]<<" ";
}

```

12. Să se afișeze cea mai lungă secvență de elemente consecutive de parități diferite.

Exemplu pentru  $n=8$  și șirul 2 4 3 3 4 7 8 2 se va afișa 3 4 7 8

**Soluție:** Această problemă poate avea o rezolvare de complexitate cubică ( $n^3$ ). Pentru fiecare secvență de elemente situate între valorile  $i$  și  $j$  ( $1 \leq i < j \leq n$ ) se actualizează dacă este posibil secvența de lungime maximă ce conține elemente consecutive de parități diferite:

```

1 pentru i = 1, n-1 executa
2   pentru j = i+1, n executa
3     lc = j-i+1; ok = true; //lc=lungimea secvenței curente
4     pentru k = j+1, i executa
5       dacă a[k] mod 2 = a[k-1] mod 2 atunci ok = false;
6     //
7     //
8     dacă ok atunci
9       lmax = lc; //lungimea secvenței maxime
10      pmax = i; //poziția de început a secvenței
11    //
12    //
13  //
14  pentru i=pmax, pmax+lmax-1 executa
15    scrie a[i];
16

```

Vom opta însă pentru o rezolvare liniară, ceea ce înseamnă că după o singură parcurgere a elementelor vectorului  $a$  am identificat secvența maximală cerută. Pentru aceasta trebuie studiate operațiile care se impun a fi efectuate la fiecare pas al parcurgerii vectorului:

1. verificarea parității elementului curent față de elementul anterior.
  - 1.1 elementul curent are paritate diferită față de elementul anterior
  - 1.2 elementul curent are aceeași paritate diferită cu elementul anterior
2. actualizarea secvenței maxime dacă este posibil

În cazul 1.1 va trebui incrementată lungimea secvenței curente de elemente de parități diferite. În situația 1.2 secvența se reinițializează, lungimea curentă  $lc$  devenind 1 iar poziția de început a secvenței va fi indicele curent  $i$ . După efectuarea uneia din aceste operații se actualizează, dacă este posibil, secvența maximală.

```

1 var a:array[1..100] of byte;
2   n,i,j,pc,pmax,lc,lmax:byte;
3 begin
4   readln(n);
5   for i:=1 to n do read(a[i]);
6   lmax:=0; lc:=1; pc:=1;
7   for i:=2 to n do begin
8     if a[i] mod 2 < a[i-1] mod 2
9       then inc(lc)
10      else begin
11        lc:=1;
12        pc:=i;
13      end;
14   if lc>lmax then begin
15     lmax:=lc;
16     pmax:=pc;
17   end;
18 end;
19 for i:=pmax to lmax+pmax-1 do
20   write(a[i], ' ');
21 end.

```

```

#include <iostream.h>
int j,i,n,lmax,pmax,lc,pc;
int a[100];
void main() {
  cin>>n;
  for(i=1;i<=n;i++) cin>>a[i];
  lmax=0; lc=1; pc=1;
  for (i=2;i<=n;i++) {
    if (a[i]%2!=a[i-1]%2)
      lc++;
    else
      { lc=1; pc=i; }
    if (lmax<lc)
      { lmax=lc; pmax=pc; }
  }
  for (i=pmax;i<=lmax+pmax-1;i++)
    cout<<a[i]<<" ";
}

```

13. Se consideră doi vectori ale căror elemente sunt ordonate crescător. Să se realizeze un algoritm prin care se realizează operația de *interclasare* a elementelor celor două tablouri.

**Soluție:** Algoritmul de interclasare parcurge elementele celor doi vectori realizând compararea succesivă a elementelor curente. Compararea începe cu elementele situate pe prima poziție, cel mai mic fiind plasat într-un nou vector ( $C$ ). Se înaintează cu o poziție în vectorul din care s-a copiat elementul plasat în  $C$ , ș.a.m.d.

```

1 var
2   a,b,c:array[1..100] of integer;
3   j,k,x,i,n,m:integer;
4 begin
5   readln(n); readln(m);
6   for i:=1 to n do read(a[i]);
7   for i:=1 to m do read(b[i]);
8   i:=1; j:=1; k:=0;
9   while (i<=n) and (j<=m) do begin
10     inc(k);
11     if a[i]<b[j] then begin
12       c[k]:=a[i]; inc(i);
13     end

```

```

#include <iostream.h>
int a[100],b[100],c[100];
int j,k,x,i,n,m;
void main() {
  cin>>n; cin>>m;
  for (i=0;i<n;i++)
    cin>>a[i];
  for (i=0;i<m;i++)
    cin>>b[i];
  i=0; j=0; k=-1;
  while (i<n && j<m)
    if (a[i]<b[j])
      c[++k]=a[i++];

```

```

14 else begin
15   c[k]:=b[j]; inc(j); end;
16 end;
17 for i:=j to m do begin
18   inc(k); c[k]:=b[i]
19 end;
20 for j:=i to n do begin
21   inc(k); c[k]:=a[j]
22 end;
23 for i:=1 to m+n do write(c[i])
end.

```

14. Realizați un program care determină toate numerele mai mici decât o valoare naturală  $N$  ( $N < 30000$ ), folosindu-se de algoritmul numit „Ciurul lui Eratostene”.

**Soluție:** Ciurul lui Eratostene „cerne” dintre toate numerele naturale mai mici ca  $N$ , pe cele care nu sunt prime. Ideea stă în eliminarea pe rând a tuturor multiplilor numerelor prime. Se poate folosi un tablou unidimensional *ok* în care la finalul procesării *ok[i]=true* (1 pentru C/C++) dacă  $i$  este prim și *false* (0) în caz contrar. Inițial toate valorile sunt considerate numere prime, deci vectorul va conține numai valoarea *true*(1). Vectorul *ok* va fi parcurs începând cu poziția 2, până la  $\sqrt{n}$  (un număr  $x$  este prim dacă nu are nici un divizor mai mic decât radical din  $x$ ).

```

1 var
2   ok:array[2..30000]of boolean;
3   i,j,n:integer;
4 begin
5   readln(n);
6   fillchar(ok,sizeof(ok),true);
7   for i:=2 to trunc(sqrt(n)) do
8     if ok[i] then begin
9       j:=2;
10      while i*j<=n do begin
11        ok[i*j]:=false;
12        inc(j);
13      end;
14    end;
15   for i:=2 to n do
16     if ok[i] then write(i,' ')
17 end.

```

15. Se consideră un vector ce conține elemente naturale din intervalul  $[c1..c2]$ , ( $c2-c1 < 10000$ ;  $c1, c2 < 30000$ ). Să se ordoneze valorile tabloului folosind sortarea prin numărarea aparițiilor fiecărui element (*Count Sort*).

**Soluție:** Datorită lungimii destul de mici a intervalului în care se regăsesc elementele, ne putem folosi de un algoritm ce are la bază determinarea numărului de apariții a fiecărei valori. Ne vom folosi de un vector *ap* în care elementul *ap[x]* indică numărul de apariții al valorii  $x + c1$  (pentru a translați indicii în domeniul  $[0..c2-c1]$ ).

Pe baza acestuia se va construi vectorul în care valorile vor fi plasate în ordine crescătoare. Algoritmul are o complexitate liniară în lungimea domeniului de definiție al elementelor.

```

1 var
2   a,ap:array[0..10000]of integer;
3   k,max,c1,c2,n,i,j,x,min:integer;
4 begin
5   readln(c1,c2,n);
6   max:=c1; min:=c2;
7   for i:=1 to n do begin
8     read(x); inc(ap[x-c1]);
9     if x>max then max:=x;
10    if x<min then min:=x
11  end;
12  k:=0; max:=max-c1; min:=min-c1;
13  for i:=min to max do
14    for j:=1 to ap[i] do begin
15      inc(k); a[k]:=i+c1;
16      write(a[k])
17    end;
18  end.

```

## 2.1.4 Probleme propuse

1. Se cunosc notele obținute de  $n$  elevi la extemporul de matematică. Să se realizeze un program care afișează:

- câte note mai mici ca 5 au fost obținute;
- care este media aritmetică a notelor peste 5.
- câte note de 7 au fost obținute.
- care este cea mai mare notă obținută;

2. Se citește un șir de  $n$  ( $n < 500$ ) numere naturale. Care este numărul maxim și de câte ori apare în cadrul șirului?

*Exemplu:*  $n=5$  și vectorul (8, 9, 6, 9, 9) se va afișa 9 apare de 3 ori

3. Cunoscându-se un șir de  $n$  numere naturale se cere realizarea unui program care permite numărarea elementelor care se divid cu ultimul element al șirului.

*Exemplu:*  $n=5$  și vectorul (8, 4, 6, 9, 3) se va afișa 2

4. Se consideră un șir de  $n$  elemente numere reale. Să se înlocuiască fiecare element cu cel mai apropiat întreg și să se afișeze în ordine inversă (de la ultimul către primul).

*Exemplu:*  $n=6$  și vectorul: (2.72, 4.34, 9.82, 1.0, 4.05, 2.45) se va afișa  
2 4 1 10 4 3

5. Se consideră un vector cu  $n$  elemente numere întregi. Să se calculeze suma tuturor elementelor pare situate pe poziții impare în tablou.  
*Exemplu:*  $n=5$  și vectorul (3, 4, 6, 7, 8) se va afișa 14

6. Se consideră un vector ce conține  $n$  cifre ( $n < 10$ ). Să se determine suma numerelor formate cu cifrele din vector citite de la dreapta la stânga și de la stânga la dreapta.

*Exemplu:*  $n=4$  și vectorul: (2, 0, 4, 5) se va afișa  $2045+5402=7447$

7. Se consideră un vector cu  $n$  elemente numere naturale. Să se înlocuiască fiecare element noul cu media aritmetică a numerelor nenule din vector.

*Exemplu:*  $n=5$  și vectorul: (2, 0, 4, 0, 3) se va afișa 2 3 4 3 3

8. Se consideră un vector cu  $n$  elemente numere naturale. Să se afișeze după fiecare element al vectorului valoarea 0.

*Exemplu:*  $n=3$  și vectorul: (2, 4, 3) se va afișa 2 0 4 0 3 0

9. Se citește un vector cu  $n$  elemente numere reale. Să se afișeze toate perechile de elemente egal depărtate de mijloc, care au aceeași parte întreagă.

*Exemplu:*  $n=6$  și vectorul: (2.32, 4.34, 9.2, 1.0, 4.05, 2.45) se va afișa 2.32 2.45 respectiv 4.34 4.05

10. Ordonăți descrescător elementele nenule ale unui vector ce conține  $n$  numere întregi.

*Exemplu:*  $n=6$  și vectorul: (3 0 7 0 4 5) se va afișa 7 0 5 0 4 3

11. Se consideră un tablou unidimensional cu  $n$  elemente numere întregi. Ștergeți toate aparițiile primului element și afișați elementele rămase.

*Exemplu:*  $n=5$  și vectorul (3, 4, 3, 3, 8) se va afișa 4 8

12. Se consideră un șir de  $n$  caractere citite de la tastatură. Care este caracterul care apare de cele mai multe ori și care este numărul de apariții?

13. Se consideră un vector cu  $n$  elemente numere reale. Inserați în fața fiecărui element negativ un element de valoare 0. Elementele vor fi afișate cu 2 zecimale.  
*Exemplu:*  $n=4$  și vectorul: (2.32, -4.34, -9.2, 1.0) se va afișa 2.32 0.00 -4.34 0.00 -9.20 1.00

14. Determinați cea mai lungă secvență de elemente pozitive din cadrul unui vector.

*Exemplu:*  $n=6$  și vectorul: (3, -4, 3, 13, 8, -3) se va afișa 3 13 8.

15. Determinați mulțimea ce se formează cu elementele unui vector.

*Exemplu:*  $n=6$  și vectorul: (3, 13, 3, 13, 8, 13) se va afișa 3 13 8.

16. Afișați pentru un șir de  $n$  elemente care este numărul de apariții al fiecărei valori. *Exemplu:*  $n=6$  și vectorul: (3, 13, 3, 13, 8, 13) se va afișa

13 3

3 2

8 1.

17. Determinați suma maximă care se poate forma cu  $m$  numere distincte dintr-un vector ce conține  $n$  valori întregi. Dacă vectorul conține mai puțin de  $m$  valori distincte se va afișa mesajul Imposibil.

*Exemplu:*  $n=6, m=2$  și vectorul: (3, 13, 3, 13, 8, 13) se va afișa 21

18. Să se scrie un program care afișează permutările circulare ale unui vector cu  $n$  elemente întregi. O permutare circulară se obține prin rotirea elementelor vectorului cu  $i$  poziții ( $i < n$ ).

*Exemplu:* Pentru  $n=4$  și vectorul (2,5,3,1) se va afișa:

5 3 1 2

3 1 2 5

1 2 5 3

19. Se consideră un vector ce conține  $n$  numere reale. Vom spune că două elemente ale sale, formează o "pereche în dezordine" dacă sunt îndeplinite simultan condițiile:

-  $i < j$

-  $a[i] > a[j]$ , unde  $1 \leq i < n$  și  $1 < j \leq n$

Să se creeze un program care afișează perechile în dezordine din vector și numărul lor.

*Exemplu:* Pentru  $n=4$  și vectorul (1, 13, 2, 4), se va afișa:

13 2

13 4

2

20. Afișați cifrele distincte ale unui număr în ordine crescătoare a numărului lor de apariții. *Exemplu:* Pentru  $n=355222$  se va afișa 3 5 2.

21. Se consideră un vector ce conține  $n$  ( $n < 100$ ) numere naturale cu valori între 0 și 60000. Să se ordoneze elementele pare, fără însă a afecta pozițiile pe care sunt situate numerele impare. Programul va afișa pe ecran, vectorul după ordonare. Evitați folosirea unui vector auxiliar. *Exemplu:* Pentru  $n=7$  și șirul (1, 40, 32, 44, 3, 8, 17), se va afișa: 1 8 32 40 3 44 17.

22. Se consideră un vector care conține  $n$  elemente de tip *char*. Creați un program care afișează perechea de două elemente egal depărtate de centru, a căror sumă a codurilor ASCII este maximă (printre celelalte perechi).

*Exemplu:* Pentru  $n=7$  și elementele 'A', 'C', 'B', 'E', 'z', 'l', 'E' se va afișa B z

23. Creați un program care sortează elementele situate între elementul minim și maxim dintr-un vector de întregi. Dacă minimul și maximum se află pe poziții consecutive, se va afișa mesajul "Nu se efectuează sortarea".

*Exemplu:* Pentru  $n=7$  și elementele 14, -13, 21, 1, 120, 1000, 21 se va afișa:  
14 -13 1 21 120 1000 21

24. Se cunosc elementele reale ale unui vector de dimensiune  $2*n$ . Să se afișeze  $n$  numere reale obținute prin adunarea părților întregi cu părțile fracționare a elementelor egal depărtate de centru. Afișarea numerelor se va face cu 3 zecimale.

*Exemplu:* Pentru  $n=3$  și numerele: 2.3, 12.09, 218.98021, 31.05, -212.098, 12.75, se va afișa: 2.750 12.098 218.050

25. Se consideră două mulțimi reținute în doi vectori. Să se realizeze un program care determină reuniunea, intersecția și diferența lor.

*Exemplu:*  $A=(2,4,1,6,7)$ ,  $B=(3,4,8,9)$  se va afișa:

$$A \cup B = (2, 4, 1, 6, 7, 3, 8, 9); A \cap B = (4); A - B = (2, 1, 6, 7)$$

26. Fie un tablou unidimensional ce conține  $n$  numere naturale în care există un singur element nul. Să se realizeze un program care ordonează descrescător elementele vectorului, efectuând interschimbări doar prin intermediul elementului nul (orice interschimbare are loc între un element nenul și cel nul)

*Exemplu:* Pentru vectorul (0,3,2) pașii sortării pot fi (3,0,2) și (3,2,0)

27. Se consideră două tablouri unidimensionale  $A$  și  $B$  ce conțin  $n$  respectiv  $m$  elemente ( $n < m$ ). Verificați dacă există în  $B$  o secvență de  $n$  elemente (situate pe poziții consecutive), nu neapărat în aceeași ordine. Dacă nu există afișați 0, altfel afișați primul indice din  $B$  de la care se regăsesc toate elementele din  $A$ .

*Exemplu:* Pentru  $n=3$ ,  $m=7$ ,  $A=(2,4,1)$  și  $B=(3,4,8,4,2,1,9)$  se va afișa: 4

28. Să se genereze primii  $n$  termeni ce fac parte din șirul definit după cum urmează:

- primul termen 1.
- dacă  $x$  aparține șirului atunci și  $2x+1$  și  $3x+1$  se regăsesc în șir;

Observație: termenii șirului nu sunt neapărat distincți.

*Exemplu:* Pentru  $n=6$  se va afișa: 1, 3, 4, 7, 10, 9

29. Un număr are forma unui "munte", dacă cifrele ce apar în scrierea lui zecimală, formează inițial un șir crescător, apoi un șir descrescător. De exemplu numărul 2556431 este un număr munte. Verificați dacă scrierea unui număr  $n$  ( $n < 2.000.000.000$ ) citit de la tastatură, respectă regula precizată.

30. Se consideră un tablou unidimensional  $A$  cu  $n$  elemente ce reprezintă o permutare a mulțimii  $1..n$ . Asupra elementelor lui se vor face următoarele tipuri de mutări: în ordine de la 1 la  $n$ , fiecare element  $a[i]$  se va interschimba cu elementul de pe poziția  $i+a[i]$ . Dacă această poziție este mai mare decât  $n$ , atunci număratoarea se continuă cu poziția 1. Să se afișeze conținutul vectorului la finalul operațiilor.

*Exemplu:* Pentru  $n=4$  și  $A=(2,4,1,3)$  se va afișa: (2, 4, 3, 1), deoarece se obține succesiv (1, 4, 2, 3); (1, 4, 2, 3); (2, 4, 1, 3); (2, 4, 3, 1)

31. Să se ștergă din vectorul  $A$  de lungime  $n$ , un număr de elemente, astfel încât la final să se obțină un șir strict crescător de elemente. Primul element din vectorul inițial nu se va șterge.

*Exemplu:* Pentru  $n=7$  și  $A=(3,4,8,4,2,1,9)$  se va afișa: (3, 4, 8, 9)

32. Se consideră un vector cu  $n$  elemente naturale. Să se afișeze pe linii, elementele din  $A$  grupate după cifra dominantă (prima în scrierea zecimală). Pe aceeași linie vor fi scrise elemente cu aceeași cifră dominantă.

*Exemplu:* Pentru  $n=7$  și  $A=(334, 124, 21, 34, 122, 1, 39)$  se va afișa:

124 122 1

21

334 34 39

33. Fie un tablou unidimensional cu  $n$  elemente valori naturale. Să se ordoneze descrescător aceste valori, după numărul de cifre distincte pe care le conțin.

*Exemplu:* Pentru  $n=7$  și  $A=(334, 124, 21, 34, 222, 1, 39)$  se va afișa:

124 334 21 34 39 222 1

34. Se consideră un vector cu  $n$  elemente numere reale. Să se ordoneze elementele crescător după valoarea părților întregi a elementelor, iar la valori cu partea întreagă egală, ordonarea se va face descrescător după partea fracționară.

*Exemplu:* Pentru  $n=7$  și  $A=(3.34, 12.4, 3.41, 3.04, 12.8, 1.3, 3.9)$  se va afișa: 1.3, 3.9, 3.41, 3.34, 3.04, 12.8, 12.4

35. Considerăm un tablou unidimensional ce conține un număr par de elemente ( $2*n$ ). Creați cu aceste valori un șir de  $n$  fracții a căror sumă este maximă. Fiecare fracție se va afișa pe câte o linie printr-o pereche de numere reprezentând în ordine „numărător - numitor”.

*Exemplu:* Pentru  $n=3$  și  $A=(3, 12, 21, 34, 2, 39)$  se va afișa:

34 3

21 12

39 2

36. Se consideră un vector ce conține  $n$  elemente întregi. În fața oricărui element precedat de un element de semn contrar se introduce un element pozitiv, a cărui valoare este obținută prin alipirea cifrelor celor două numere de semne contrare, în ordine. Să se afișeze conținutul vectorului după efectuarea operațiilor cerute.

*Exemplu:* Pentru  $n=6$  și  $A=(3, -1, 73, 5, -9, 2)$  se va afișa: 3, 31, -1, 173, 73, 5, 59, -9, 92, 2

37. Fie un tablou unidimensional cu  $2*n$  elemente valori naturale. Din vector sunt șterse pe rând elemente din  $k$  în  $k$  poziții. Numărarea pozițiilor de va face cu revenire la prima în cazul în care indicele curent este mai are decât  $n$ . Operația se repetă de  $n$  ori. Determinați poziția de început a numărătorii, astfel încât elementele rămase să aibă suma maximă. Se va afișa poziția de început a numărătorii și suma elementelor rămase în vector.

*Exemplu:* Pentru  $n=3$ ,  $k=3$  și  $A=(3, 10, 15, 4, 2, 10)$  se va afișa: 2 35. Începând numărătoarea de la poziția 2, au fost șterse în ordine elementele de pe pozițiile 4, 1, 5.

38. Se consideră un tablou unidimensional cu  $n(<100)$  elemente naturale. Să se determine numărul minim de subșiruri strict crescătoare de valori consecutive în care poate fi partiționat vectorul. Prin subșir se înțelege o secvență de elemente din vectorul inițial ce nu se află neapărat pe poziții consecutive.

*Exemplu:* Pentru  $n=7$  și  $A=(3, 10, 4, 4, 5, 11, 6)$  se va afișa: 3 (3 4 5 6; 10 11; 4)

39. Se consideră un tablou unidimensional cu  $n(<100)$  elemente întregi. Să se determine toate secvențele de elemente de pe poziții consecutive, care au suma egală cu  $S$ . Fiecare secvență de elemente va fi afișată pe câte o linie pe ieșirea standard.

*Exemplu:* Pentru  $n=7$ ,  $S=9$  și vectorul (3, 2, 3, 4, 5, 11, -7) se va afișa:

2 3 4

4 5

5 11 -7

40. Fie un tablou unidimensional cu  $n$  elemente valori naturale. Să se determine o submulțime de elemente din tabloul citit, pentru care suma elementelor este divizibilă cu  $n$ .

*Exemplu:* Pentru  $n=7$  și  $A=(3, 6, 4, 2, 11, 5, 11)$  se va afișa: 6, 4, 2, 11, 5

41. Se consideră doi vectori de lungime  $n$  respectiv  $m$  ce conține elemente naturale ordonate crescător. Se cere interclasarea valorilor pare din cei doi vectori. În urma interclasării elementele vor fi plasate într-un nou vector.

*Exemplu:* Pentru  $n=8$ ,  $m=5$ ,  $A=(13, 26, 44, 54, 112, 115, 311, 600)$   $B=(3, 28, 48, 55, 56)$  se va afișa: 26, 28, 44, 48, 54, 56, 112, 600

42. Fie un tablou unidimensional cu  $2^n$  elemente valori naturale. Din acesta se poate obține un alt vector  $B$  cu  $2^{n-1}$  elemente astfel: Elementul  $b[1]$  se va fi egal cu produsul  $a[1] * a[2]$ ,  $b[2]$  cu produsul  $a[3] * a[4]$  ș.a.m.d. Operația se poate aplica în continuare și asupra vectorului  $B$ . Practic, ea se poate efectua succesiv de cel mult  $n$  ori asupra vectorului obținut la pasul precedent. Scrieți un program care execută operația descrisă cât timp toate elementele obținute sunt mai mici ca un număr  $S$  (citit de la intrarea standard). Afișați elementele obținute în final în vector.

*Exemplu:* Pentru  $n=3$ ,  $S=200$  și  $A=(1, 2, 4, 5, 2, 5, 3, 6)$  se va afișa 40 180. Din vectorul inițial s-a obținut vectorul (2, 20, 10, 18), iar la pasul următor (40 180).

43. Se consideră un tablou unidimensional cu  $n(<100)$  elemente întregi (pozitive/negative). Avem la dispoziție  $K(<n)$  semne minus (-) pe care vom folosi pentru schimbarea semnelor la  $K$  elemente din vector. Nu se pot folosi două semne minus pentru un singur element. Determinați produsul maxim care se poate obține cu  $n-1$  elemente din vector după ce s-au schimbat semnele la  $k$  elemente și valorile care au fost înmulțite.

*Exemplu:* Pentru  $n=7$ ,  $k=6$  și  $A=(-2, -4, 5, -2, 6, 3, 6)$  se afișează 4320.

$4 * (-5) * 2 * (-6) * (-3) * (-6)$ .

44. Se consideră șirul următor: 1, 3, 4, 7, 8, 10, 11, 15, 16, 18, 19, 22, 23, 25, 26, 31, 32.... Să se determine al  $n$ -lea termen al șirului.

45. Se citește de la tastatură valorile din doi vectori. Primul conține  $n$  elemente în ordine crescătoare, iar al doilea  $m$  elemente în ordine descrescătoare. Să se realizeze un program care interclasează doar elementele impare din cei doi vectori și afișează pe ecran șirul obținut.

*Exemplu:* Pentru  $n=8$ ,  $m=5$ ,  $A=(13, 26, 44, 54, 112, 115, 311, 600)$   $B=(567, 55, 48, 5, 3)$  se va afișa: 3, 5, 13, 55, 115, 311, 567

46. Se citește de la tastatură  $n$  elemente ale unui vector. Să se realizeze un program care șterge elementele situate între prima apariție a unui element cub perfect și ultima apariție a unui element cub perfect.

*Exemplu:* Pentru  $n=8$  și  $A=(13, 26, 28, 54, 112, 8, 2197, 600)$  se va afișa:  $A=(13, 26, 600)$

47. Se consideră un șir de  $n$  valori numere naturale și  $m$  perechi de indici ( $i, j$ ) unde  $0 < i < j < n+1$ . Elementele situate între  $i$  și  $j$  își inversează pozițiile, ca în exemplu de mai jos. Să se afișeze ordinea elementelor din vector la finalul operațiilor.

*Exemplu:* Pentru  $n=8$ ,  $A=(13, 26, 28, 54, 11, 78, 21, 60)$ ,  $m=2$  și perechile (1,8) și (2,5) după prima operație de rotire vectorul arată:

60, 21, 78, 11, 54, 28, 26, 13;

După a doua operație de rotire, între pozițiile 2 și 5, vectorul conține:

60, 54, 11, 78, 21, 28, 26, 13.

## 2.2. Tabloul bidimensional

### 2.2.1 Teste cu alegere multiplă și duală

1. Considerăm declarația `var a:array[1..10,1..10] of byte` (varianta Pascal), respectiv `int a[10][10]` (varianta C/C++). Identificați care din următoarele accesări de elemente ale variabilei `a` sunt incorecte:

- |                            |                             |
|----------------------------|-----------------------------|
| a) <code>a[2*3]</code>     | a) <code>a[2*3]</code>      |
| b) <code>a[2*3,3-2]</code> | b) <code>a[2*3][3-2]</code> |
| c) <code>a[10],[1]</code>  | c) <code>a[10,1]</code>     |
| d) <code>a[10,10]</code>   | d) <code>a[9][9]</code>     |

2. Care din următoarele declarații sunt incorecte sintactic?

- |  |                                      |
|--|--------------------------------------|
| a) <code>a:array[1..10][1..10] of char</code>  | a) <code>char a[1..10][1..10]</code> |
| b) <code>a:array[2..10;3..10] of real</code>   | b) <code>float a[2..10;3..10]</code> |
| c) <code>a:array['a'..'z',1..5] of real</code> | c) <code>float a[26][5]</code>       |
| d) <code>a:array[1..5,6..9] of char</code>     | d) <code>char [5][9]</code>          |

3. Care dintre următoarele variante reprezintă declarația unui tablou bidimensional cu 5 linii și 5 coloane ale cărui elemente sunt de tip `real`?

- |  |                                |
|--|--------------------------------|
| a) <code>a:array[0..5,0..5] of real</code>     | a) <code>float a[6][6]</code>  |
| b) <code>a:array[10..14,1..5] of real</code>   | b) <code>float a[5][5]</code>  |
| c) <code>a:array['a'..'e',1..5] of real</code> | c) <code>double a[5][5]</code> |
| d) <code>a:array['0'..'5',1..5] of char</code> | d) <code>char a[6][5]</code>   |

4. Considerăm următoarele declarații:

```
var a:array[0..4,0..4] of byte; unsigned char a[5][5];
    b:array['a'..'e',1..5] of real; float b[5][5];
```

Care dintre variantele următoare accesează corect un element din cele două tablouri situat pe linia a treia și coloana a doua?

- |                            |                             |
|----------------------------|-----------------------------|
| a) <code>a[2,3]</code>     | a) <code>a[2][3]</code>     |
| b) <code>a[3,2]</code>     | b) <code>a[3][2]</code>     |
| c) <code>b[3,2]</code>     | c) <code>b['c'][2]</code>   |
| d) <code>b['c',2]</code>   | d) <code>b[2][1]</code>     |
| e) <code>a[2,1]</code>     | e) <code>a[2][1]</code>     |
| f) <code>b['c','b']</code> | f) <code>b['c']['b']</code> |

5. Considerăm un tablou cu  $n$  linii și  $m$  coloane în care toate elementele primei linii sunt nule. Știind că nu există alt element egal cu zero (nesituat pe prima linie), câte elemente nenule sunt în tablou?

- |              |              |            |            |
|--------------|--------------|------------|------------|
| a) $(m+n)-n$ | b) $(m+n)-m$ | c) $n*m-n$ | d) $n*m-m$ |
|--------------|--------------|------------|------------|

6. Considerăm un tablou cu  $n$  linii și  $m$  coloane. Câte elemente sunt situate pe marginea tabloului (prima și ultima linie, prima și ultima coloană)?

- |              |                  |                |                |
|--------------|------------------|----------------|----------------|
| a) $2*m+2*n$ | b) $m*n-2*n-2*m$ | c) $2*n+2*m-4$ | d) $2*n+2*m-2$ |
|--------------|------------------|----------------|----------------|

7. Care dintre următoarele variante calculează în mod corect numărul de elemente nule de pe fiecare linie a unei matrici pătratice de ordin  $n$ ? Afișarea rezultatelor trebuie făcută începând cu prima linie.

- |   |  |
|---|--|
| a) <pre>nr:=0; for i:=1 to n do begin   for j:=1 to n do     if a[i,j]=0 then inc(nr);   write(nr) end;</pre>           | a) <pre>nr:=0; for (i:=0;i&lt;n;i++){   for (j:=0;j&lt;n;j++){     if (a[i][j]==0) nr++;   }   cout &lt;&lt; nr;</pre>     |
| b) <pre>for i:=1 to n do begin   nr:=0;   for j:=1 to n do     if a[i,j]=0 then inc(nr);   write(nr) end;</pre>         | b) <pre>for (i:=0;i&lt;n;i++) {   nr:=0;   for (j:=0;j&lt;n;j++)     if (a[i][j]==0) nr++;   cout &lt;&lt; nr;</pre>       |
| c) <pre>for i:=n downto 1 do begin   nr:=0;   for j:=n downto 1 do     if a[i,j]=0 then inc(nr);   write(nr) end;</pre> | c) <pre>for (i:=n-1;i&gt;=0;i--) {   nr:=0;   for (j:=n-1;j&gt;=0;j--)     if (a[i][j]==0) nr++;   cout &lt;&lt; nr;</pre> |
| d) <pre>for i:=1 to n do begin   nr:=0;   for j:=n downto 1 do     if a[i,j]=0 then inc(nr);   write(nr) end;</pre>     | d) <pre>for (i:=0;i&lt;n;i++) {   nr:=0;   for (j:=n-1;j&gt;=0;j--)     if (a[i][j]==0) nr++;   cout &lt;&lt; nr;</pre>    |

8. Considerăm declarația: `var a: array[1..3,1..3] of byte`; (varianta Pascal), respectiv `int a[3][3]`; (varianta C/C++). Specificați care va fi conținutul tabloului după execuția secvenței următoare de instrucțiuni:

<pre>for i:=1 to 3 do   for j:=1 to 3 do a[i,j]:=i+j;</pre>	<pre>for (i:=0;i&lt;3;i++)   for (j:=0;j&lt;3;j++) a[i][j]=i+j+2;</pre>														
a) <table border="0"><tr><td>2 2 2</td><td>4 4 4</td><td>6 6 6</td></tr></table>	2 2 2	4 4 4	6 6 6	b) <table border="0"><tr><td>2 3 4</td><td>2 3 4</td><td>2 3 4</td></tr></table>	2 3 4	2 3 4	2 3 4	c) <table border="0"><tr><td>2 2 2</td><td>3 3 3</td><td>4 4 4</td></tr></table>	2 2 2	3 3 3	4 4 4	d) <table border="0"><tr><td>2 3 4</td><td>3 4 5</td><td>4 5 6</td></tr></table>	2 3 4	3 4 5	4 5 6
2 2 2	4 4 4	6 6 6													
2 3 4	2 3 4	2 3 4													
2 2 2	3 3 3	4 4 4													
2 3 4	3 4 5	4 5 6													

9. Considerăm declarația `var a: array[1..3,1..3] of byte`;  $i, j$ : byte (varianta Pascal), respectiv `int i, j, a[3][3]`; (varianta C/C++). Specificați care va fi conținutul tabloului după execuția secvenței următoare de instrucțiuni:

<pre>for i:=1 to 3 do   for j:=1 to 3 do     a[i,j]:=abs(i-j);</pre>	<pre>for (i:=0;i&lt;3;i++)   for (j:=0;j&lt;3;j++)     a[i][j]=abs(i-j);</pre>
--	--

a) 1 1 1      b) 1 1 1      c) 0 1 2      d) 1 2 3  
 2 2 2      1 1 1      1 0 1      2 1 2  
 3 3 3      1 1 1      2 1 0      3 2 1

10. Considerăm declarația *var a : array[1..3,1..3] of byte; i, j: byte* (varianta Pascal), respectiv *int i, j, a[3][3];* (varianta C/C++). Care sunt instrucțiunile necesare pentru ca tabloul a să conțină elementele:

1 2 3  
 4 5 6  
 7 8 9

a) for i:=1 to 3 do  
 for j:=1 to 3 do  
 a[i,j]:=abs(i-j);

b) for i:=1 to 3 do  
 for j:=1 to 3 do  
 a[i,j]:= i+j;

c) for i:=1 to 3 do  
 for j:=1 to 3 do  
 a[i,j]:=i\*j;

d) for i:=1 to 3 do  
 for j:=1 to 3 do  
 a[i,j]:= (i-1)\*3+j;

a) for (i=0;i<3;i++)  
 for (j=0;j<3;j++)  
 a[i][j]:=abs(i-j+2);

b) for (i=0;i<3;i++)  
 for (j=0;j<3;j++)  
 a[i][j]:=i+j+2;

c) for (i=0;i<3;i++)  
 for (j=0;j<3;j++)  
 a[i][j]=(i+1)\*(j+1);

d) for (i=0;i<3;i++)  
 for (j=0;j<3;j++)  
 a[i][j]=i\*3+j+1;

11. Care dintre declarațiile următoare reprezintă declarația unui tablou bidimensional cu 10 linii și 20 coloane și componente de tip real:

a) type matrice:  
 array[1..10,1..20] of real;  
 var mat=matrice;

b) var mat:array[1..10,1..20] of  
 real;

c) var mat:array[1..20,1..10] of  
 integer;

d) var mat:array[1..10,1..20] of  
 extended;

a) typedef float m[10][20] matrice;  
 matrice mat;

b) float mat[10][20];

c) int mat[20][10];

d) double mat[10][20];

12. Care dintre secvențele de mai jos realizează interschimbarea a două linii, *l1* respectiv *l2*, ale unei matrice cu *n* linii și *m* coloane de numere întregi?

a) for j:=1 to m do  
 a[l1,j]:=a[l2,j];

b) for j:=1 to n do  
 a[l2,j]:=a[l1,j];

a) for (j=0;j<m;j++)  
 a[l1][j]=a[l2][j];

b) for (j=0;j<n;j++)  
 a[l2][j]=a[l1][j];

c) for j:=1 to n do begin  
 aux:=a[l1,j];  
 a[l1,j]:=a[l2,j];  
 a[l2,j]:=aux;  
 end;

d) for j:=1 to m do begin  
 aux:=a[l1,j];  
 a[l1,j]:=a[l2,j];  
 a[l2,j]:=aux;  
 end;

c) for (j=0;j<n;j++) {  
 aux=a[l1][j];  
 a[l1][j]=a[l2][j];  
 a[l2][j]=aux;  
 }  
 )

d) for (j=0;j<m;j++) {  
 aux=a[l1][j];  
 a[l1][j]=a[l2][j];  
 a[l2][j]=aux;  
 }  
 )

13. O matrice pătratică este simetrică față de diagonala principală dacă pentru orice pereche de indici (*i, j*):

a) a[i,j]=-a[j,i];  
 b) a[i,j]=1/a[i,j];  
 c) a[i,j]=a[j,i];  
 d) a[i,j]<>a[i,j];

a) a[i][j]==-a[j][i];  
 b) a[i][j]==1/a[i][j];  
 c) a[i][j]==a[j][i];  
 d) a[i][j]!=a[i][j];

14. Se consideră următoarea secvență de program în care *a* este o matrice pătratică cu *n* linii și *n* coloane, iar *i, j, k* și *l* sunt variabile de tip întreg:

```
for i:=1 to n do
  for j:=1 to n do read(a[i,j]);
i:=2;
j:=n-1;
for l:=1 to n div 2 do begin
  for k:=i to j do
    write(a[l,k], ' ');
  writeln;
  inc(i);
  dec(j);
end;
```

```
for (i=1;i<=n;i++)
  for (j=1;j<=n;j++) cin>>a[i][j];
i=2;
j=n-1;
for (l=1;l<=n/2;l++) {
  for (k=i;k<=j;k++)
    cout<<a[l][k]<<" ";
  cout<<endl;
  i++;
  j--;
}
```

Secvența de mai sus afișează:

- elementele matricei *a* aflate atât strict sub diagonala secundară, cât și strict sub diagonala principală;
- elementele matricei *a* aflate strict deasupra diagonalei secundare;
- elementele matricei *a* aflate atât strict deasupra diagonalei secundare, cât și strict deasupra diagonalei principale;
- elementele matricei *a* aflate strict deasupra diagonalei principale.

15. Se consideră următoarea secvență de program în care *a* este o matrice pătratică cu *n* linii și *n* coloane, iar *i, j* și *k* sunt variabile de tip întreg:

```
k:=-1;
for i:=1 to n do begin
  if k=-1 then
    for j:=1 to n do
      write(a[i,j], ' ');
```

```
k=-1;
for (i=1;i<=n;i++) {
  if (k=-1)
    for (j=1;j<=n;j++)
      cout<<a[i][j]<<" ";
```

```

else
  for j:=n downto 1 do
    write(2*a[i,j], ' ');
    k:=k*(-1);
  end;

```

```

else
  for (j:=n; j>=1; j--)
    cout<<2*a[i][j]<<" ";
    k*=-1;
  }

```

Știind că după executarea secvenței de program de mai sus au fost afișate valorile 1 2 3 8 6 4 3 4 5 și că matricea pătratică  $a$  are 3 linii și 3 coloane, stabiliți care dintre tablourile de mai jos reprezintă matricea  $a$ .

a)	b)	c)	d)
3 2 1	1 2 3	1 2 3	1 2 3
8 6 4	5 6 7	2 3 4	2 3 4
5 4 3	3 4 5	3 4 5	5 4 3

16. Se consideră următoarea secvență de program în care  $a$  este o matrice pătratică cu  $n$  linii și  $n$  coloane, iar  $i$  și  $j$  sunt variabile de tip întreg:

```

for i:=1 to 4 do
  for j:=1 to 4 do
    if i<=j then a[i,j]:=i
    else a[i,j]:=j;
  for (i:=1; i<=4; i++)
    for (j:=1; j<=4; j++)
      if (i<=j) a[i][j]:=i;
      else a[i][j]:=j;

```

Stabiliți care dintre variantele de mai jos reprezintă matricea  $a$  obținută după executarea secvenței de mai sus:

a)	b)	c)	d)
1 1 1 1	1 1 1 1	1 2 3 4	1 1 1 1
2 2 2 2	1 2 2 2	1 2 3 3	2 2 2 1
3 3 3 3	1 2 3 3	1 2 2 2	3 3 3 2
4 4 4 4	1 2 3 4	1 1 1 1	4 3 2 1

17. Dacă  $a$  este un tablou unidimensional de numere întregi și  $n$  lungimea sa, iar  $aux$  este o variabilă de tip întreg ce efect are secvența următoare:

```

for i:=1 to n-1 do
  if (a[i]>a[i+1]) begin
    aux:=a[i];
    a[i]:=a[i+1];
    a[i+1]:=aux;
  end;
  for (i:=0; i+1<n; i++)
    if (a[i]>a[i+1]) {
      aux:=a[i];
      a[i]:=a[i+1];
      a[i+1]:=aux;
    }

```

- Ordonează crescător elementele tabloului
- Interschimbă elementele tabloului astfel încât la final cel mai mare element se afle pe ultima poziție a tabloului
- Interschimbă elementele tabloului astfel încât la final cel mai mic element se afle pe prima poziție a tabloului
- Interschimbă elementele de pe poziții pare cu cele de pe poziții impare

18. Se consideră declarațiile:

```

var a:array[0..9,0..9] of integer;
x:array[0..99] of integer;
int a[10][10], x[100];

```

Știm că vectorul  $x$  a fost folosit pentru liniarizarea matricei  $a$ . Care dintre elementele din  $x$  următoare, reprezintă elementul de pe linia  $i$  și coloana  $j$  din matricea  $a$ :

- $x[i * 10 + j - 1]$
- $x[(i-1) * 10 + j - 1]$
- $x[i * 10 + j]$
- $x[(i-1) * 10 + j]$

19. Care din următoarele variante reprezintă o declarație corectă a unui tablou unidimensional cu 10 de componente de tip caracter:

- |                                  |                     |
|----------------------------------|---------------------|
| a) var mat:array[1..10] of byte; | a) char mat[1..10]; |
| b) var mat:array[1..10] of char; | b) char mat[10];    |
| c) var sir[1..10] of char;       | c) mat[10] of char; |
| d) var sir=array[1..10] of char; | d) char[10];        |

### 2.2.2 Teste cu itemi semiobiectivi

1. Se consideră următorul program pseudocod:

```

1  întreg i, n, x, a[10][10];
2  citește n, m;
3  pentru i ← 1, n executa
4  | pentru j ← 1, m executa
5  | | citește a[i,j];
6  |
7  |
8  pentru i ← 1, n executa
9  | x ← 0;
10 | pentru j ← 1, m executa
11 | | x ← x + a[i,j] mod 10;
12 |
13 | scrie x;
14 | stop.

```

a) Ce se va afișa pentru  $n=2$ ,  $m=3$  și tabloul  $A$  cu elementele citite în ordine pe linii: 12, 40, 51, 44, 654, 33?

b) Dați un exemplu de set de date de intrare pentru care se va afișa un șir de valori crescătoare.

c) Dați exemplu de set de date de intrare pentru care se va afișa un un șir de valori reprezentând suma elementelor pe fiecare linie.

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

2. Se consideră următorul program pseudocod:

```

1  citește n, m;
2  pentru i ← 1, n executa
3  | pentru j ← 1, m executa
4  | | citește a[i,j];
5  |
6  |
7  max ← 0;

```

a) Ce se va afișa pentru  $n=3$ ,  $m=3$  și tabloul  $A$  cu elementele citite în ordine pe linii: 15, 40, 51, 44, 12, 33, 2, 33, 5?



```

8  pentru j ← 1, m executa
9      x ← 0;
10     pentru i ← 1, n executa
11         daca a[i,j] mod 3 = 0 atunci
12             x ← x + 1;
13     ■
14     ■
15     ■
16     ■
17     ■
18     ■
19     ■
20     ■
21     ■
22     ■
23     ■
24     ■
25     ■
26     ■
27     ■
28     ■
29     ■
30     ■
31     ■
32     ■
33     ■
34     ■
35     ■
36     ■
37     ■
38     ■
39     ■
40     ■
41     ■
42     ■
43     ■
44     ■
45     ■
46     ■
47     ■
48     ■
49     ■
50     ■
51     ■
52     ■
53     ■
54     ■
55     ■
56     ■
57     ■
58     ■
59     ■
60     ■
61     ■
62     ■
63     ■
64     ■
65     ■
66     ■
67     ■
68     ■
69     ■
70     ■
71     ■
72     ■
73     ■
74     ■
75     ■
76     ■
77     ■
78     ■
79     ■
80     ■
81     ■
82     ■
83     ■
84     ■
85     ■
86     ■
87     ■
88     ■
89     ■
90     ■
91     ■
92     ■
93     ■
94     ■
95     ■
96     ■
97     ■
98     ■
99     ■
100    ■

```

b) Care este valoarea maximă pe care o poate lua variabila *max*? Dați un exemplu de set de date de intrare pentru această situație.

c) Dați exemplu de set de date de intrare pentru care se va afișa valoarea 0.

d) Modificați algoritmul astfel încât să afișeze și numărul coloanei pe care s-a obținut valoarea finală a variabilei *max*.

e) Realizați programul în limbajul de programare studiat Pascal/C/C++, pentru algoritmul de la punctul d)

3. Se consideră următorul program pseudocod:

```

1  întreg i, n, linie, x,
2      max, a[10][10];
3  citește n, m;
4  pentru i ← 1, n executa
5      pentru j ← 1, m executa
6          citește a[i,j];
7      ■
8      ■
9      max ← 0;
10     linie ← 0;
11     pentru i ← 1, n executa
12         x ← 0;
13         pentru j ← 2, m executa
14             daca a[i,j] = a[i,1] atunci
15                 x ← x + 1;
16         ■
17         ■
18         ■
19         ■
20         ■
21         ■
22         ■
23         ■
24         ■
25         ■
26         ■
27         ■
28         ■
29         ■
30         ■
31         ■
32         ■
33         ■
34         ■
35         ■
36         ■
37         ■
38         ■
39         ■
40         ■
41         ■
42         ■
43         ■
44         ■
45         ■
46         ■
47         ■
48         ■
49         ■
50         ■
51         ■
52         ■
53         ■
54         ■
55         ■
56         ■
57         ■
58         ■
59         ■
60         ■
61         ■
62         ■
63         ■
64         ■
65         ■
66         ■
67         ■
68         ■
69         ■
70         ■
71         ■
72         ■
73         ■
74         ■
75         ■
76         ■
77         ■
78         ■
79         ■
80         ■
81         ■
82         ■
83         ■
84         ■
85         ■
86         ■
87         ■
88         ■
89         ■
90         ■
91         ■
92         ■
93         ■
94         ■
95         ■
96         ■
97         ■
98         ■
99         ■
100        ■

```

a) Ce se va afișa pentru  $n=3, m=4$  și tabloul *A* cu elementele citite în ordine pe linii: 7, 7, 3, 2, 34, 3, 3, 2, 2, 2, 4, 2?

b) Care este valoarea maximă pe care o poate lua *max*? Dați un exemplu de set de date de intrare pentru această situație.

c) Dați exemplu de set de date de intrare pentru care se vor afișa valorile 0 0.

d) Modificați algoritmul astfel încât să afișeze și produsul elementelor situate pe linia pe care s-a obținut valoarea finală a variabilei *max*.

e) Realizați programul în limbajul de programare studiat Pascal/C/C++, pentru algoritmul de la punctul d)

4. Se consideră următorul program pseudocod:

```

1  întreg i, n, j, a[10][10];
2  citește n;
3  pentru j ← 1, n executa
4      pentru i ← 1, n executa
5          a[i,j] ← j;
6      ■
7      ■
8      ■
9      ■
10     ■
11     ■
12     ■
13     ■
14     ■
15     ■
16     ■
17     ■
18     ■
19     ■
20     ■
21     ■
22     ■
23     ■
24     ■
25     ■
26     ■
27     ■
28     ■
29     ■
30     ■
31     ■
32     ■
33     ■
34     ■
35     ■
36     ■
37     ■
38     ■
39     ■
40     ■
41     ■
42     ■
43     ■
44     ■
45     ■
46     ■
47     ■
48     ■
49     ■
50     ■
51     ■
52     ■
53     ■
54     ■
55     ■
56     ■
57     ■
58     ■
59     ■
60     ■
61     ■
62     ■
63     ■
64     ■
65     ■
66     ■
67     ■
68     ■
69     ■
70     ■
71     ■
72     ■
73     ■
74     ■
75     ■
76     ■
77     ■
78     ■
79     ■
80     ■
81     ■
82     ■
83     ■
84     ■
85     ■
86     ■
87     ■
88     ■
89     ■
90     ■
91     ■
92     ■
93     ■
94     ■
95     ■
96     ■
97     ■
98     ■
99     ■
100    ■

```

a) Ce se va afișa pentru  $n=4$ ?  
b) Ce se va afișa dacă instrucțiunea:  $a[i,j] \leftarrow j$  devine  $a[i,j] \leftarrow i$ ?  
c) Modificați algoritmul astfel încât elementele în cadrul unei linii să fie egale cu numărul liniei respective.  
d) Realizați programul în limbajul de programare studiat Pascal/C/C++, pentru algoritmul de la punctul c).  
e) Modificați algoritmul astfel încât elementele matricei să fie completate, în ordine, pe coloane cu primele  $n^2$  numere pare.

5. Se consideră următorul program pseudocod:

```

1  întreg i, n, m, j, a[10][10];
2  citește n, m; k ← 0;
3  pentru i ← 1, n executa
4      daca i mod 2 = 1 atunci
5          pentru j ← 1, m executa
6              k ← k + 1; a[i,j] ← k;
7          ■
8          altfel
9              pentru j ← m, 1, -1 executa
10                  k ← k + 1; a[i,j] ← k;
11          ■
12          ■
13          ■
14          ■
15          ■
16          ■
17          ■
18          ■
19          ■
20          ■
21          ■
22          ■
23          ■
24          ■
25          ■
26          ■
27          ■
28          ■
29          ■
30          ■
31          ■
32          ■
33          ■
34          ■
35          ■
36          ■
37          ■
38          ■
39          ■
40          ■
41          ■
42          ■
43          ■
44          ■
45          ■
46          ■
47          ■
48          ■
49          ■
50          ■
51          ■
52          ■
53          ■
54          ■
55          ■
56          ■
57          ■
58          ■
59          ■
60          ■
61          ■
62          ■
63          ■
64          ■
65          ■
66          ■
67          ■
68          ■
69          ■
70          ■
71          ■
72          ■
73          ■
74          ■
75          ■
76          ■
77          ■
78          ■
79          ■
80          ■
81          ■
82          ■
83          ■
84          ■
85          ■
86          ■
87          ■
88          ■
89          ■
90          ■
91          ■
92          ■
93          ■
94          ■
95          ■
96          ■
97          ■
98          ■
99          ■
100         ■

```

a) Ce se va afișa dacă la intrare vor fi introduse valorile:  $n=4$  și  $m=3$ ?  
b) Ce se va afișa dacă instrucțiunea  $a[i,j] \leftarrow k$  devine  $a[i,j] \leftarrow n*m-k+1$ ?  
c) Modificați algoritmul astfel încât elementele tabloului să fie completate pe coloane după aceeași regulă prezentată alăturat.  
d) Ce se va afișa dacă instrucțiunea pentru de pe linia 9 devine pentru  $j \leftarrow 1, m$  executa  
e) Realizați programul în limbajul de programare studiat Pascal/C/C++, pentru algoritmul de la punctul c).

6. Se consideră următorul program pseudocod:

```

1  întreg i, n, j, m, a[100][100];
2  citește n, m;
3  pentru i ← 1, n executa
4      pentru j ← 1, m executa
5          a[i,j] ← i;
6      ■
7      ■
8      ■
9      ■
10     ■
11     ■
12     ■
13     ■
14     ■
15     ■
16     ■
17     ■
18     ■
19     ■
20     ■
21     ■
22     ■
23     ■
24     ■
25     ■
26     ■
27     ■
28     ■
29     ■
30     ■
31     ■
32     ■
33     ■
34     ■
35     ■
36     ■
37     ■
38     ■
39     ■
40     ■
41     ■
42     ■
43     ■
44     ■
45     ■
46     ■
47     ■
48     ■
49     ■
50     ■
51     ■
52     ■
53     ■
54     ■
55     ■
56     ■
57     ■
58     ■
59     ■
60     ■
61     ■
62     ■
63     ■
64     ■
65     ■
66     ■
67     ■
68     ■
69     ■
70     ■
71     ■
72     ■
73     ■
74     ■
75     ■
76     ■
77     ■
78     ■
79     ■
80     ■
81     ■
82     ■
83     ■
84     ■
85     ■
86     ■
87     ■
88     ■
89     ■
90     ■
91     ■
92     ■
93     ■
94     ■
95     ■
96     ■
97     ■
98     ■
99     ■
100    ■

```

a) Ce valori vor fi afișate pentru  $n=4$  și  $m=4$ ? Dar pentru  $n=4$  și  $m=1$ ?  
b) Dați un exemplu pentru datele de intrare, astfel încât numărul de elemente nenule din matrice să fie egal cu numărul de elemente nule.  
c) Determinați în funcție de  $n$  și  $m$  care este numărul de valori nule?  
d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

7. Se consideră următorul program pseudocod:

```

1  intreg i, n, j, s, a[100][100];
2  citește n;
3  s ← 0;
4  pentru i ← 1, n executa
5  | pentru j ← 1, n executa
6  | | citește a[i,j];
7  |
8  |
9  | pentru i ← 1, (n+1) div 2 exec.
10 | | pentru j ← i, n-i+1 executa
11 | | | s ← s + a[i,j];
12 | |
13 |
14 scrie s.
15

```

a) Ce se va afișa pentru  $n=4$  și tabloul  $A$  cu elementele citite în ordine pe linii: 1, 2, 3, 4, 4, 3, 2, 1, 1, 2, 3, 4, 4, 3, 2, 1, ?

b) Determinați în funcție de  $n$  pentru câte elemente din matrice a fost calculată suma?

c) Modificați instrucțiunile de pe liniile 9 și 10 astfel încât să fie calculată suma elementelor dispuse simetric față de linia de mijloc a tabloului?

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

8. Se consideră următorul program pseudocod:

```

1  intreg i, n, j, p, a[100][100];
2  citește n;
3  p ← 1;
4  pentru i ← 1, n executa
5  | pentru j ← 1, n executa
6  | | citește a[i,j];
7  |
8  |
9  | pentru j ← 1, (n+1) div 2 exec.
10 | | pentru i ← j, n-j+1 executa
11 | | | p ← p * a[i,j];
12 | |
13 |
14 scrie p.
15

```

a) Ce se va afișa pentru  $n=4$  și tabloul  $A$  cu elementele citite în ordine pe linii: 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, ?

b) Determinați în funcție de  $n$  pentru câte elemente din matrice a fost calculat produsul?

c) Modificați instrucțiunile de pe liniile 9 și 10 astfel încât să fie calculată suma elementelor dispuse simetric față de coloana de mijloc a tabloului?

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

9. Se consideră următorul program pseudocod:

```

1  citește n; m ← 0;
2  pentru i ← 1, n executa
3  | pentru j ← 1, n-i+1 executa
4  | | m ← m + 1;
5  | | a[i,j] ← m;
6  |
7  |

```

a) Care vor fi elementele tabloului  $A$  pentru  $n=4$ ?

b) Determinați o valoare a lui  $n$  astfel încât elementele să fie ordonate crescător pe linii?

```

8  pentru i ← n, 2, -1 executa
9  | pentru j ← n, n-i+2, -1 exec.
10 | | m ← m + 1;
11 | | a[i,j] ← m;
12 |
13 |
14 |
15 |

```

c) Modificați limitele de ciclare ale instrucțiunii de pe linia 5 astfel încât elementelor de pe diagonala secundară să nu li se atribuie nici o valoare?

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

10. Se consideră următorul program pseudocod:

```

1  intreg i, n, j, m, a[100][100];
2  citește n;
3  pentru i ← 1, n executa
4  | pentru j ← 1, n executa
5  | | a[i,j] ← |i-j|+1;
6  |
7  |
8  | pentru i ← 1, n executa
9  | | pentru j ← 1, n executa
10 | | | scrie a[i,j];
11 | |
12 |
13 |

```

a) Care vor fi elementele tabloului  $A$  pentru  $n=4$ ?

b) Determinați o valoare a lui  $n$  astfel încât elementele pe linii să formeze șiruri monotone crescătoare sau descrescătoare?

c) Modificați instrucțiunea de atribuire de pe linia 5 astfel încât, completând elementele după aceeași regulă, cea mai mică valoare din matrice să fie 0.

d) Realizați programul în limbajul de programare studiat Pascal/C/C++.

### 2.2.3 Probleme rezolvate

1. Completați elementele unui tablou bidimensional pătratic de ordin  $n$ , sub forma unor pătrate concentrice de valori consecutive, începând cu 1.

Exemplu: Pentru  $n=4$  elementele tabloului vor fi:

```

1 1 1 1
1 2 2 1
1 2 2 1
1 1 1 1

```

**Soluție:** Trebuie traversate un număr de  $[(n+1)/2]$  pătrate concentrice. Algoritmul va parcurge succesiv pătratele, traversând simultan cele două linii (latura de sus – latura de jos) respectiv cele două coloane (latura stângă – latura dreaptă).

```

1  var
2  | a:array[1..10,1..10] of byte;
3  | n,i,j,r:integer;
#include <iostream.h>
unsigned char a[10][10];
int n,i,j,r;

```

```

4 begin
5   readln(n);
6   for r:=1 to (n+1)div 2 do begin
7     for j:=r to n-r+1 do begin
8       a[r,j]:=r; a[n-r+1,j]:=r;
9     end;
10    for i:=r+1 to n-r do begin
11      a[i,r]:=r; a[i,n-r+1]:=r;
12    end;
13  end;
14  for i:=1 to n do begin
15    for j:=1 to n do write(a[i,j]);
16    writeln
17  end;
18 end.

void main() {
  cin>>n;
  for (r=1;r<=(n+1)/2;r++) {
    for (j=r;j<=n-r+1;j++) {
      a[r-1][j-1]=r;
      a[n-r][j-1]=r;
    }
    for (i=r+1;i<=n-r;i++) {
      a[i-1][r-1]=r; a[i-1][n-r]=r;
    }
  }
  for (i=0;i<n;i++) {
    for (j=0;j<n;j++)
      cout<<(int)a[i][j];
    cout<<endl; } }

```

2. Se consideră un tablou bidimensional cu  $n$  linii și  $m$  coloane. Să se determine numărul de perechi de linii monotone. Două linii se numesc monotone dacă oricare pereche de elemente ale lor situate pe aceeași coloană respectă monotonia elementelor de pe prima coloană.

*Exemplu:*

Pentru  $n=3, m=4$

și matricea  $A$ :

8 6 7 6

2 1 4 7

4 2 3 5

se va afișa:

1

Linile 1 și 3 formează o pereche de linii monotone.

*Soluție:* Algoritmul testează pentru orice pereche de două linii dacă sunt monotone. Pentru aceasta se memorează în variabila  $mt$  relația de monotonie dintre primele două elemente. Toate celelalte  $m-1$  perechi de elemente de pe cele două linii trebuie să aibă aceeași monotonie.

```

1 var
2   a:array[1..10,1..10]of byte;
3   n,l,i,j,nr,m:integer;
4   mt,ok:boolean;
5 begin
6   readln(n,m);
7   nr:=0;
8   for i:=1 to n do
9     for j:=1 to m do read(a[i,j]);
10  for l:=1 to n-1 do
11    for i:=l+1 to n do begin
12      mt:=a[l,1]<a[i,1];
13      ok:=true;
14      for j:=2 to m do
15        if (a[l,j]<a[i,j])<>mt then
16          ok:=false;
17      if ok then inc(nr);
18    end;
19  writeln(nr);
20 end.

#include <iostream.h>
unsigned char a[10][10];
int n,l,i,j,nr,m,mt,ok;
void main() {
  cin>>n>>m; nr=0;
  for (i=0;i<n;i++)
    for (j=0;j<m;j++) cin>>a[i][j];
  for (l=0;l+1<n;l++)
    for (i=l+1;i<n;i++) {
      mt=a[l][0]<a[i][0]; ok=1;
      for (j=1;j<m;j++)
        if ((a[l][j]<a[i][j])!=mt)
          ok=0;
      if (ok) nr++;
    }
  cout<<nr;
}

```

3. Se consideră un tablou bidimensional ce memorează pe fiecare din cele  $n$  linii câte o mulțime de  $m$  elemente. Să se realizeze un program care determină perechea de mulțimi cu intersecție de cardinal maxim. Se vor afișa numerele de ordine ale liniilor din pereche.

*Exemplu:*

Pentru  $n=4, m=4$

și matricea  $A$ :

8 6 7 9

2 1 4 5

9 2 7 8

4 2 3 7

se va afișa:

1 3

*Soluție:* Algoritmul identifică perechea de linii cu intersecție maximă în  $O(n^4)$ . Pentru oricare două linii, toate elementele de pe una dintre ele sunt căutate pe cealaltă, contorizându-se numărul de elemente comune.

```

1 var
2   a:array[1..10,1..10]of byte;
3   n,l,i,j,nr,m,max,c,x,y,t:byte;
4 begin
5   readln(n,m); max:=0;
6   for i:=1 to n do
7     for j:=1 to m do read(a[i,j]);
8   for i:=1 to n-1 do
9     for j:=i+1 to n do begin
10      nr:=0;
11      for c:=1 to m do
12        for t:=1 to m do
13          if a[i,c]=a[j,t] then inc(nr);
14          if nr>max then begin
15            max:=nr; x:=i; y=j;
16          end;
17      end;
18  writeln(x,' ',y); end.

#include <iostream.h>
int a[10][10],n,l,i,j,nr,m,max,c,x,y,t;
void main() {
  cin>>n>>m;
  max=0;
  for (i=0;i<n;i++)
    for (j=0;j<m;j++) cin>>a[i][j];
  for (i=0;i+1<n;i++)
    for (j=i+1;j<n;j++) {
      nr=0;
      for (c=0;c<m;c++)
        for (t=0;t<m;t++)
          if (a[i][c]==a[j][t]) nr++;
      if (nr>max) {max=nr;x=i;y=j;}
    }
  cout<<x+1<<' '<<y+1<<endl;
}

```

4. Se consideră un tablou bidimensional ce memorează pe fiecare din cele  $n$  linii câte o mulțime de  $m$  elemente. Să se realizeze un program care determină perechea de mulțimi cu reuniunea de cardinal maxim. Se va afișa numerele de ordine ale liniilor din pereche.

*Exemplu:*

Pentru  $n=4, m=4$

și matricea  $A$ :

8 6 7 9

2 1 4 5

9 2 7 8

4 2 3 7

se va afișa:

1 2

**Soluție:** Algoritmul identifică perechea de linii cu reuniune maximă în  $O(n^4)$ . Pentru oricare două linii, se consideră că reuniunea lor are cardinalul egal cu  $2 * m$ , adică nu au elemente comune. Orice element comun care este identificat micșorează cardinalul reuniunii cu o unitate.

```

1 var
2   a:array[1..10,1..10]of byte;
3   n,l,i,j,nr,m,max,c,x,y,t:byte;
4 begin
5   readln(n,m); max:=0;
6   for i:=1 to n do
7     for j:=1 to m do read(a[i,j]);
8   for i:=1 to n-1 do
9     for j:=i+1 to n do begin
10      nr:=2*m;
11      for c:=1 to m do
12        for t:=1 to m do
13          if a[i,c]=a[j,t] then dec(nr);
14      if nr>max then begin
15        max:=nr; x:=i; y:=j;
16      end;
17    end;
18  writeln(x,' ',y);
19 end.

```

```

#include <iostream.h>
int a[10][10],n,l,i,j,nr,m,max,
c,x,y,t;
void main() {
  cin>>n>>m;
  max=0;
  for (i=0;i<n;i++)
    for (j=0;j<m;j++)
      cin>>a[i][j];
  for (i=0;i+1<n;i++)
    for (j=i+1;j<n;j++) {
      nr=2*m;
      for (c=0;c<m;c++)
        for (t=0;t<m;t++)
          if (a[i][c]==a[j][t])nr--;
      if (nr>max){max=nr;x=i;y=j;}
    }
  cout<<x+1<<' '<<y+1<<endl;
}

```

5. Fie un tablou bidimensional de  $n$  linii și  $m$  coloane. Să se ștergă toate liniile care încep cu un element ce se regăsește pe prima linie. Valorile elementelor sunt naturale mai mici decât 1000. **Exemplu:**

Pentru  $n=4$ ,  $m=4$  și matricea  $A$  :

8 6 7 9	8 6 7 9
7 1 4 5	4 2 3 7
9 2 7 8	
4 2 3 7	

**Soluție:** Algoritmul folosește un vector suplimentar  $s$  cu indici mai mici ca 1000, ale cărui elemente sunt 0 sau 1. Astfel,  $s[i]=1$  dacă valoarea  $i$  se găsește pe prima linie în matrice.

Ștergerea unei linii  $x$  se face prin deplasarea tuturor liniilor  $x+1, \dots, n$  cu o poziție mai sus. Valoarea variabilei  $i$  care indică linia curentă în prelucrare, se incrementează cu o unitate, numai dacă linia respectivă nu a fost ștearsă. Fiecare operație de ștergere este însoțită de o decrementare a numărului de linii a tabloului.

```

1 var
2   a:array[1..10,1..10]of byte;
3   s:array[0..1000]of byte;
4   n,l,i,j,nr,m:integer;
5 begin
6   readln(n,m);
7   for i:=1 to n do
8     for j:=1 to m do read(a[i,j]);
9   for i:=1 to m do s[a[1,i]]:=1;

```

```

#include <iostream.h>
int a[10][10],s[1001];
int n,l,i,j,nr,m;
void main() {
  cin>>n>>m;
  for (i=0;i<n;i++)
    for (j=0;j<m;j++) cin>>a[i][j];
  for (i=0;i<m;i++) s[a[0][i]]=1;
  i=1;

```

```

10 i:=2;
11 while i<=n do
12   if s[a[i,1]]=1 then begin
13     for l:=i+1 to n do
14       for j:=1 to m do
15         a[l-1,j]:=a[l,j];
16       dec(n); end
17   else inc(i);
18   for i:=1 to n do begin
19     writeln;
20     for j:=1 to m do write(a[i,j]);
21   end; end.

```

```

while (i<n)
  if (s[a[i][0]]==1) {
    for (l=i+1;l<n;l++)
      for (j=0;j<m;j++)
        a[l-1][j]=a[l][j];
    n--;
  }
  else i++;
  for (i=0;i<n;i++) {
    cout<<endl;
    for (j=0;j<m;j++)
      cout<<a[i][j]<<' ';
  }
}

```

6. Fie un tablou bidimensional de  $n$  linii și  $m$  coloane. Să se realizeze un program care inserează în fața oricărei linii ale cărei elemente sunt ordonate crescător sau descrescător, o nouă linie cu elemente egale cu valoarea maximă de pe linia ordonată.

**Exemplu:** se va afișa : 8 8 8 8  
 Pentru  $n=2$ ,  $m=4$  1 4 5 8  
 și matricea  $A$ : 9 9 9 9  
 1 4 5 8  
 9 8 5 2

**Soluție:** Inserarea unei linii  $i$  se face prin deplasarea tuturor liniilor începând cu  $n$ ,  $n-1, \dots, i$  cu o poziție mai jos. Valoarea variabilei  $i$  care indică linia curentă în prelucrare, se incrementează cu o unitate dacă nu s-a efectuat inserarea unei linii, sau cu două unități dacă s-a inserat o nouă linie pe poziția. Fiecare operație de inserare este însoțită de o incrementare a numărului de linii a tabloului.

```

1 var
2   a:array[1..10,1..10]of byte;
3   ok,w:boolean;
4   n,l,i,j,m,c,v:integer;
5 begin
6   readln(n,m);
7   for i:=1 to n do
8     for j:=1 to m do read(a[i,j]);
9   i:=1;
10  while i<=n do begin
11    ok:=a[i,1]<a[i,2];
12    w:=true;
13    for j:=2 to m-1 do
14      if (a[i,j]<a[i,j+1])<>ok then
15        w:=false;
16    if w then begin
17      for l:=n downto i do
18        for c:=1 to m do
19          a[l+1,c]:=a[l,c];
20      if ok then v:=a[i,m]
21      else
22        v:=a[i,1];

```

```

#include <iostream.h>
unsigned char a[10][10];
int ok,w,n,l,i,j,m,c,v;
void main() {
  cin>>n>>m;
  for (i=0;i<n;i++)
    for (j=0;j<m;j++)
      cin>>a[i][j];
  i=0;
  while (i<n) {
    ok=a[i][0]<a[i][1]; w=1;
    for (j=1;j+1<m;j++)
      if ((a[i][j]<a[i][j+1])!=ok)
        w=0;
    if (w) {
      for (l=n-1;l>=i;l--)
        for (c=0;c<m;c++)
          a[l+1][c]=a[l][c];
      if (ok) v=a[i][m-1];
      else v=a[i][0];
      for (c=0;c<m;c++) a[i][c]=v;
      n++;
    }
    i++;
  }
}

```

```

23   for c:=1 to m do a[i,c]:=v;   i:=2;
24   inc(n);                       }
25   inc(i,2)                       else i++;
26   end                             }
27   else inc(i);                   }
28   end;                           }
end.

```

7. Se consideră un tablou bidimensional  $A(n,m)$  cu elemente întregi. Realizați un program care inversează elementele tabloului, prin intermediul unui vector de  $n*m$  elemente:

*Exemplu:*

Pentru  $n=3, m=3$  se va afișa: 9 8 7  
și matricea  $A$ : 6 5 4  
1 2 3 3 2 1  
4 5 6  
7 8 9

*Soluție:* Algoritmul liniarizează matricea folosind vectorul auxiliar  $v$ . Astfel, oricare element al tabloului bidimensional  $a[i][j]$  se va regăsi în vector pe poziția  $(i-1)*m+j$ .

```

1 var
2   a:array[1..10,1..10]of byte;
3   v:array[1..100]of byte;
4   n,x,i,j,m:integer;
5 begin
6   readln(n,m);
7   for i:=1 to n do
8     for j:=1 to m do
9       read(a[i,j]);
10    for i:=1 to n do
11      for j:=1 to m do
12        v[(i-1)*m+j]:=a[i,j];
13    for i:=1 to m*n div 2 do begin
14      x:=v[i];
15      v[i]:=v[m*n-i+1];
16      v[m*n-i+1]:=x;
17    end;
18    for i:=1 to n do
19      for j:=1 to m do
20        a[i,j]:=v[(i-1)*m+j];
21    for i:=1 to n do begin
22      writeln;
23      for j:=1 to m do write(a[i,j]);
24    end;
25  end;
26 end.

```

```

#include <iostream.h>
unsigned char a[10][10],v[100];
int n,x,i,j,m;
void main() {
  cin>>n>>m;
  for (i=0;i<n;i++)
    for (j=0;j<m;j++)
      cin>>a[i][j];
  for (i=0;i<n;i++)
    for (j=0;j<m;j++)
      v[i*m+j]=a[i][j];
  for (i=0;i<(m*n)/2;i++) {
    x=v[i]; v[i]=v[m*n-i-1];
    v[m*n-i-1]=x;
  }
  for (i=0;i<n;i++)
    for (j=0;j<m;j++)
      a[i][j]=v[i*m+j];
  for (i=0;i<n;i++) {
    cout<<endl;
    for (j=0;j<m;j++)
      cout<<a[i][j];
  }
}

```

8. Se consideră un tablou bidimensional  $A(n,m)$  cu elemente în mulțimea cifrelor 0..9. Realizați un program care determină pentru fiecare linie, baza minimă în care cifrele respective pot reprezenta un număr. Considerând că pe linie este descris în această bază un număr, determinați valoarea obținută la conversia lui în baza 10. Pentru fiecare linie se va afișa baza minimă și valoarea după conversie.

*Exemplu:* Pentru  $n=3, m=3$   
și matricea  $A$ :  
1 2 3  
1 0 1  
7 8 9

se va afișa:  
4 27  
2 5  
10 789

*Soluție:* În scrierea unui număr în baza  $b$  se folosesc cifrele 0.. $b-1$ . Algoritmul identifică elementul maxim de pe fiecare linie, iar baza minimă corectă este mai mare cu o unitate decât acesta.

```

1 var
2   a:array[1..10,1..10]of byte;
3   n,b,i,j,m,nr:integer;
4 begin
5   readln(n,m);
6   for i:=1 to n do
7     for j:=1 to m do read(a[i,j]);
8   for i:=1 to n do begin
9     b:=a[i,1];
10    nr:=0;
11    for j:=2 to m do
12      if a[i,j]>b then b:=a[i,j];
13    inc(b);
14    for j:=1 to m do
15      nr:=nr*b+a[i,j];
16    writeln(b,' ',nr);
17  end;
18 end.

```

```

#include <iostream.h>
int a[10][10];
int n,b,i,j,m,nr;
void main() {
  cin>>n>>m;
  for (i=0;i<n;i++)
    for (j=0;j<m;j++)
      cin>>a[i][j];
  for (i=0;i<n;i++) {
    b=a[i][0]; nr=0;
    for (j=1;j<m;j++)
      if (a[i][j]>b) b=a[i][j];
    b++;
    for (j=0;j<m;j++)
      nr=nr*b+a[i][j];
    cout<<b<<' '<<nr<<endl;
  }
}

```

9. La un concurs de patinaj ce conține  $m$  probe, participă  $n$  sportivi identificați prin numere de la 1 la  $n$  ( $n,m < 100$ ). Să se afișeze numerele de ordine ale sportivilor ce au obținut cele mai mari 3 punctaje totale. Punctajele concurenților sunt introduse de la tastatură, în ordinea numerelor de concurs. Nota maximă a unei probe este 10.

*Exemplu:* Pentru  $n=5, m=3$  și punctajele: 5, 2, 3; 5, 5, 5; 1 0 1; 7 8 9; 4 4 2 se va afișa:  
Premiul 1: 4; Premiul 2: 2; Premiul 3: 1, 5.

*Soluție:* Se vor plasa într-o matrice de două coloane, pentru fiecare concurent, numărul de identificare și suma punctajelor obținute de acesta. Matricea va fi ordonată descrescător după coloana punctajelor (a doua).

```

1 var
2 a:array[1..100,1..2] of byte;
3 n,b,i,j,x,m,p:integer;
4 begin
5 readln(n,m);
6 for i:=1 to n do begin
7   a[i,2]:=0;
8   a[i,1]:=i;
9   for j:=1 to m do begin
10    read(x);
11    inc(a[i,2],x);
12   end;
13 end;
14 for i:=1 to n-1 do
15   for j:=i+1 to n do
16     if a[i,2]<a[j,2] then begin
17       x:=a[i,2];
18       a[i,2]:=a[j,2];
19       a[j,2]:=x;
20       x:=a[i,1];
21       a[i,1]:=a[j,1];
22       a[j,1]:=x;
23     end;
24 write(a[1,1]);
25 p:=1;
26 i:=2;
27 while (i<=n) and (p<=3) do begin
28   if (a[i,2]<>a[i-1,2]) then
29     begin inc(p);writeln;end;
30   if p<4 then write(a[i,1]);
31   inc(i);
32 end.

```

```

#include <iostream.h>
int a[100][2];
int n,b,i,j,x,m,p;
void main() {
  cin>>n>>m;
  for (i=0;i<n;i++) {
    a[i][1]=0;
    a[i][0]=i;
    for (j=0;j<m;j++)
      { cin>>x;
        a[i][1]+=x; }
  }
  for (i=0;i<n;i++)
    for (j=i+1;j<n;j++)
      if (a[i][1]<a[j][1]) {
        x=a[i][1];
        a[i][1]=a[j][1];
        a[j][1]=x;
        x=a[i][0];
        a[i][0]=a[j][0];
        a[j][0]=x; }
  cout<<a[0][0]+1<<' '; p=0; i=1;
  while (i<n && p<3) {
    if (a[i][1]!=a[i-1][1])
      { p++;
        cout<<endl; }
    if (p<3) cout<<a[i][0]+1<<' ';
    i++;
  } }

```

## 2.2.4 Probleme propuse

1. Se consideră un tablou bidimensional cu  $n$  linii și  $m$  coloane ce conține numere naturale. Realizați un program care determină suma elementelor de pe fiecare linie cu număr de ordine par și produsul elementelor de pe fiecare coloană cu număr de ordine impar.

2. Se consideră un tablou bidimensional cu  $n$  linii și  $n$  coloane ce conține numere naturale. Realizați un program care determină elementul maxim de pe diagonala principală a matricei și linia pe care acesta este situat.

3. Scrieți un program care completează elementele unui tablou pătratic de ordin  $n$  astfel:

- elementele diagonalei principale sunt egale cu 0;
- elementele situate sub diagonala principală sunt egale cu 1
- elementele situate deasupra diagonalei principale sunt egale cu 2

Exemplu: Pentru  $n=3$  se va afișa:

```

0 2 2
1 0 2
1 1 0

```

4. Se consideră două tablouri bidimensionale de dimensiuni identice ( $n \times m$ ). Să se afișeze transpusa matricei sumă. Transpusa unei matrice se obține prin schimbarea liniilor cu coloanele.

Exemplu: Pentru  $n=3$ ,  $m=2$  și Se va afișa:

tablourile:

2 3	5 6	7 4 4
3 4	1 1	9 5 9
4 9	0 0	

5. Realizați un program care determină numărul liniei cu cele mai multe elemente pare, al unei matrice pătratică de dimensiune  $n \times n$ . Dacă există mai multe linii cu număr maxim de elemente pare se va afișa una singură.

Exemplu: Pentru  $n=3$  și matricea:

se va afișa 2.

```

1 2 130
4 150 6
7 8 900

```

6. Se consideră un tablou bidimensional cu  $n$  linii și  $m$  coloane. Realizați un program care identifică linia cu cele mai multe elemente divizibile cu primul element situat pe ea.

Exemplu: Pentru  $n=3$ ,  $m=3$  și matricea:

se va afișa 2.

```

2 2 135
3 150 6
7 8 900

```

7. Realizați un program care determină cel mai mare divizor comun al elementelor situate pe fiecare coloană, a unei matrice pătratică.

Exemplu: Pentru  $n=3$  și matricea:

se va afișa 1 2 15.

```

2 4 135
3 160 15
7 6 30

```

8. Realizați un program care ordonează crescător doar elementele pare, situate pe liniile cu număr de ordine par, al unui tablou bidimensional cu  $n$  linii și  $m$  coloane.

Exemplu: Pentru  $n=3$ ,  $m=4$  și matricea:

se va afișa

2 4 13 2	2 4 13 2
24 1 6 12	6 1 12 24
16 7 30 2	16 7 30 2

9. Realizați un program care ordonează descrescător elementele de pe prima linie a unui tablou bidimensional numai prin operația de interschimbare a coloanelor.

*Exemplu:* Pentru  $n=3$ ,  $m=4$  și matricea:

2 4 13 2	se va afișa
3 1 60 13	13 4 2 2
16 7 30 2	60 1 3 13
	30 7 16 2

10. Realizați un program care permută circular liniile unui tablou bidimensional cu  $n$  linii și  $m$  coloane, cu o poziție mai sus:

*Exemplu:* Pentru  $n=3$ ,  $m=4$  și matricea:

2 4 13 2	se va afișa
3 1 60 13	3 1 60 13
16 7 30 2	16 7 30 2
	2 4 13 2

11. Se consideră un tablou bidimensional pătratic cu  $n$  linii. Să se determine c.m.m.d.c al valorilor ce reprezintă suma elementelor de sub diagonala principală și suma elementelor de deasupra diagonalei principale.

*Exemplu:* Pentru  $n=3$  și matricea:

1 6 0	se va afișa 3
9 1 6	
9 9 1	

12. Se consideră un tablou bidimensional pătratic cu  $n$  linii. Să se determine elementele care sunt situate pe linii și coloane de sumă egală. Un element  $a[i,j]$  va fi afișat dacă suma pe linia  $i$  este egală cu suma pe coloana  $j$ .

13. Se consideră un tablou bidimensional pătratic cu  $n$  linii. Să se determine toate elementele ce reprezintă puncte 'șă' (element minim pe linie și maxim pe coloana pe care este situat).

14. Se consideră o matrice pătratică de  $n$  linii. Să se afișeze suma elementelor situate pe cele două diagonale alăturate celei principale.

*Exemplu:* Pentru  $n=4$  și matricea:

1 2 3 4	se va afișa 12 (6+6)
2 1 2 3	
3 2 1 2	
4 3 2 1	

15. Să se afișeze toate elementele dintr-o matrice de  $n$  linii și  $m$  coloane care au toți vecinii numere pare. Elementele vecine lui  $a[i,j]$  sunt  $a[i-1,j]$ ,  $a[i,j-1]$ ,  $a[i+1,j]$  și  $a[i,j+1]$ , dacă există.

16. Se consideră o matrice pătratică de  $n$  linii. Să se șteargă toate liniile din tablou care încep cu un număr divizibil cu 10.

17. Să se determine elementul cu număr maxim de apariții al unui tablou bidimensional cu  $n$  linii și  $m$  coloane.

*Exemplu:* Pentru  $n=3$ ,  $m=4$  și matricea:

2 4 13 2	se va afișa '4 apare de 5 ori'
3 4 60 4	
16 4 30 4	

18. Să se determine mulțimea formată din elementele distincte de pe marginea unui tablou bidimensional pătratic.

*Exemplu:* Pentru  $n=3$  și matricea:

2 4 3	se va afișa 1 2 3 4 6
3 4 6	
1 4 3	

19. Se consideră un vector de  $n*m$  caractere. Să se completeze o matrice de  $n$  linii și  $m$  coloane cu codurile ASCII asociate caracterelor respective.

*Exemplu:* Pentru  $n=3$ ,  $m=4$  și vectorul

'A','B','C','D','a','b','c','d','A','a','B','b'	se va afișa matricea:
	65 66 67 68
	97 98 99 100
	65 97 66 98

20. Realizați un program care afișează o matrice pătratică de ordin  $n$  ale cărei elemente sunt numerele de la 1 la  $n^2$ , completate în ordine începând cu prima linie.

*Exemplu:* Pentru  $n=3$  se va afișa:

1 2 3
4 5 6
7 8 9

21. Se consideră un tablou bidimensional cu  $n$  linii și  $m$  coloane ( $1 \leq m, n \leq 100$ ) având elemente întregi. Să se determine numărul de linii care au toate componentele egale.

*Exemplu:* pentru  $n=5$ ,  $m=3$  și tabloul de mai jos, se va afișa:

7 6 3	2
4 4 4	
3 8 3	
6 6 6	
4 5 6	

22. Se consideră un tablou bidimensional cu  $m$  linii și  $n$  coloane ( $1 \leq m, n \leq 100$ ) având ca elemente cifre binare. Fiecare linie reprezintă un număr în baza 2. Se cere să se afișeze aceste numere convertite în baza 10. *Exemplu:* Pentru  $m=5$ ,  $n=4$  și tabloul:

0 0 1 1	se va afișa:
1 0 0 1	3 9 14 5 0
1 1 1 0	
0 1 0 1	
0 0 0 0	

23. Se consideră un tablou bidimensional cu  $n$  linii și  $n$  coloane ( $1 \leq n \leq 100$ ) având componente de tip întreg. Cele două diagonale ale tabloului împart tabloul în patru regiuni în formă de triunghi. Se cere să se determine suma componentelor din interiorul fiecărei zone.

Exemplu: Pentru  $n=5$  și tabloul:

0 1 1 1 0	se va afișa:
2 0 1 0 3	Suma(z1)=4
2 2 0 3 3	Suma(z2)=8
2 0 4 0 3	Suma(z3)=12
0 4 4 4 0	Suma(z4)=16

24. Se consideră un tablou bidimensional cu  $n$  linii și  $m$  coloane ( $1 \leq m, n \leq 10$ ) având componente cifre zecimale. Fiecare linie a tabloului reprezintă cifrele unui număr natural în baza 10. Se cere să se afișeze pe același rând cifrele sumei celor  $n$  numere descrise prin tabloul anterior, despărțite prin virgulă.

Exemplu: Pentru  $n=4$ ,  $m=4$  și tabloul

0 9 1 4	se va afișa:
9 2 1 1	1,3,6,9,5
3 5 4 7	
0 0 2 3	

25. Fie a o matrice cu  $n$  linii și  $m$  coloane. Scrieți un program care verifică dacă există elemente  $a_{ij}$  cu proprietatea că sunt egale cu c.m.m.m.c dintre suma elementelor de pe linia  $i$  și produsul elementelor de pe coloana  $j$ .

26. Se consideră un tablou bidimensional cu  $n$  linii și  $m$  coloane ( $1 \leq m, n \leq 50$ ) având componente numere întregi. Se cere să se afișeze liniile din tablou care au cele mai multe componente egale.

Exemplu: Pentru  $n=4$ ,  $m=5$  și tabloul:

5 2 8 4 5	se va afișa:
8 5 9 5 5	8 5 9 5 5
7 9 7 2 2	6 8 6 1 6
6 8 6 1 6	

27. Se consideră un vector cu  $n*m$  elemente de tip char. Știm că el reprezintă forma liniarizată a unei matrice de dimensiune  $n \times m$ . Să se completeze și afișeze elementele matricei, ținând cont de această presupunere.

Exemplu: pentru  $n=3$ ,  $m=2$  și vectorul ('p', 'v', 'c', 'd', 'b', 'a'), afișarea pe linii a elementelor matricei este:

p v c  
d b a

28. Să se rearanjeze elementele unei matrice de dimensiune  $n \times m$ , astfel încât ele să fie ordonate crescător atât pe linii cât și pe coloane.

Exemplu:  $n=3$ ,  $m=4$  și matricea

3 1 8 9	se va afișa:
4 6 5 7	0 1 1 2
2 0 1 3	3 3 4 5
	6 7 8 9

29. Realizați un program care afișează elementele unei matrice de dimensiune  $n \times n$ , parcurse în spirală. Exemplu: Pentru  $n=3$  și matricea:

0 1 2	se va afișa: 0 1 2 3 4 5 6 7
7 8 3	
6 5 4	

30. Creați un program care afișează elementele unei matrice pătratică de dimensiune  $n \times n$ , după ștergerea elementelor situate pe diagonala principală.

Exemplu: Pentru  $n=4$  și matricea:

0 1 1 2	se va afișa:
3 3 4 5	1 1 2
6 7 8 9	3 4 5
7 8 9 5	6 7 9
	7 8 9

31. Realizați un program care completează elementele unui tablou bidimensional  $A(n, m)$  cu valori consecutive pe linii, începând de la numărul de ordine al liniei respective.

Exemplu: Pentru  $n=3$  și  $m=4$  elementele tabloului  $A$  vor fi :

1 2 3 4  
2 3 4 5  
3 4 5 6

32. Se consideră o matrice  $A(n, m)$  ce conține numere întregi. Ordonăți crescător elementele pare situate pe ultima coloană, prin interschimbări de linii.

Exemplu: Pentru  $n=3$ ,  $m=4$  și matricea

A:	se va afișa :
1 2 3 4	3 4 5 2
2 3 4 5	1 2 3 4
3 4 5 2	2 3 4 5

33. Fie un tablou bidimensional  $A(n, m)$ . Realizați un program care inversează elementele de pe liniile care încep cu un număr prim.

Exemplu: Pentru  $n=3$ ,  $m=4$  și matricea

A:	se va afișa :
4 2 3 7	4 2 3 7
2 3 4 5	5 4 3 2
3 4 5 2	2 5 4 3

34. Determinați numerele de ordine ale liniilor unui tablou bidimensional  $A(n, m)$  care conțin cele mai multe valori palindrom.



*Exemplu:* Pentru  $n=3$ ,  $m=4$  și matricea  
 11 21 33 43  
 22 3 414 52  
 3 24 535 2  
 se va afișa :  
 2 3

35. Fie un tablou bidimensional cu  $n$  linii și  $m$  coloane cu elemente numere naturale mai mici decât 10000. Un element din tablou are ca vecini, elementele situate în imediata vecinătate pe verticală și orizontală. Să se identifice două elemente din matrice care au proprietatea că produsul vecinilor lor reprezintă cele mai mari două valori ce se pot obține. Pentru fiecare element determinat se va afișa valoarea acestuia și a produsului elementelor vecine.

*Exemplu:* Pentru  $n=4$ ,  $m=4$  și tabloul:  
 0 8 5 8  
 8 2 8 5  
 3 8 3 0  
 0 2 6 2  
 se va afișa :  
 2 4096  
 5 512

36. Fie un tablou bidimensional pătratic de ordin  $n$  cu elemente naturale. Se consideră un traseu ce pleacă din matrice de pe linia  $x$  și coloana  $y$ . Direcția de mișcare ne este dată de un șir de  $p$  caractere N, V, E, S care indică direcția de deplasare. Determinați suma elementelor situate pe drum. Elementul de start aparține drumului.

*Exemplu:*  
 Pentru  $n=4$ ,  $m=4$ ,  $x=3$ ,  $y=2$ ,  $p=6$ ,  
 traseul N,N,E,S,V,V și tabloul :  
 0 9 5 8  
 3 2 1 5  
 3 8 3 0  
 0 2 6 2  
 se va afișa 30 ( $8+2+9+5+1+2+3$ )

37. Fie un tablou bidimensional pătratic de ordin  $n$ . Considerăm un traseu ce pleacă din matrice de pe linia  $x$  și coloana  $y$ . Direcția de mișcare ne este indicată de un șir de  $p$  caractere N, V, E, S care reprezintă direcția de deplasare. Determinați elementele prin care s-a trecut de cele mai multe ori. De la intrarea standard se va prelua  $n$ ,  $x$ ,  $y$ ,  $p$  și traseul urmat. Pentru fiecare element din soluție va afișa linia și coloana pe care este situat.

*Exemplu:*  
 Pentru  $n=4$ ,  $x=3$ ,  $y=2$ ,  $p=8$  și  
 traseul N, N, E, S, V, S, S, V  
 2 2  
 3 2

38. Se consideră o tablă de șah cu  $n$  linii și  $m$  coloane, pe care sunt plasate pioni. Pionii sunt codificați la citire prin valoarea 1. Regina adversă trebuie plasată într-un punct al tablei astfel încât pe cele două diagonale pe care le atacă, să se afle câți mai mulți pioni.

Determinați linia și coloana pe care se va așeza regina și numărul de pioni de pe diagonale atacate.

*Exemplu:*  
 Pentru  $n=5$ ,  $m=4$ , tabloul :  
 1 1 0 0  
 0 1 0 1  
 1 0 1 0  
 1 0 0 0  
 1 0 1 0  
 se va afișa 4 2 5 (linia 4, coloana 2, 5 pioni)

39. Fie un tablou bidimensional cu  $n$  linii și  $m$  coloane, cu elemente întregi. Din punctul de linia  $x$  și coloana  $y$  se poate părăsi tabloul mergând numai pe orizontală sau verticală, dar numai dacă elementul este  $a[x,y]$  este negativ. Pe traseul urmat nu trebuie să se întâlnească alt element negativ. Să se determine linia și coloana de unde se poate părăsi matricea, astfel încât suma elementelor întâlnite pe traseu să fie minimă. Afișați linia și coloana punctului de start și suma elementelor de pe drum.

*Exemplu:*  
 Pentru  $n=4$ ,  $m=6$  și tabloul :  
 50 60 90 50 60 60  
 40 30 -9 -3 40 70  
 40 -1 -8 5 2 1  
 80 80 80 80 90 20  
 se va afișa 3 3 8 (linia 3, coloana 3, suma 8)

## 2.3. Fișiere text

### 2.3.1 Teste cu alegere multiplă și duală

1. Care dintre următoarele variante realizează deschiderea la citire a fișierului text 'A.TXT'

a) reset(f); assign(f, 'A.txt')	a) f=fopen("A.txt", "w");
b) assign(f, 'A.txt'); rewrite(f)	b) f=fopen("r", "A.txt");
c) assign(f, 'A.txt'); reset(A.txt)	c) f=fopen("A.txt", "r");
d) assign(f, 'A.txt'); reset(f)	d) f=fopen("A.txt", "r");

2. Care dintre următoarele variante realizează deschiderea la scriere a fișierului text 'B.TXT'

a) rewrite(f); assign(f, 'B.txt')	a) f=fopen("B.txt", "r");
b) assign(f, 'B.txt'); rewrite(f);	b) f=fopen("B.txt", "w");
c) assign(f, 'B.txt'); rewrite(B);	c) f=fopen("w", "B.txt");
d) assign(f, B.txt); rewrite(f);	d) f=fopen("B.txt", "w");

3. Care dintre următoarele instrucțiuni au ca efect citirea unui caracter din fișierul 'C.TXT'.

- |  |  |
|--|--|
| <p>a) assign(f, 'C.txt'); reset(f); close(f);</p> <p>b) assign(f, 'C.txt'); reset(f); read(x); close(f);</p> <p>c) assign(f, 'C.txt'); reset(f); read(f,x); close(f);</p> <p>d) assign(f, 'C.txt'); reset(f); close(f); read(f,x);</p> | <p>a) f=fopen("C.txt", "r"); fclose(f);</p> <p>b) f=fopen("C.txt", "r"); scanf("%c", &amp;x); fclose(f);</p> <p>c) f=fopen("C.txt", "r"); fscanf(f, "%c", &amp;x); fclose(f);</p> <p>d) f=fopen("C.txt", "r"); fclose(f); fscanf(f, "%c", &amp;x);</p> |
|--|--|

4. Considerând că variabila *f* este de tip text/fișier, care dintre următoarele instrucțiuni verifică în mod corect dacă s-a ajuns la finalul fișierului indicat de el.

- |   |   |
|---|---|
| <p>a) if eoln(f) then write('Final de fișier') else write('Nu')</p> <p>b) if eof(f) then write('Final de fișier') else write('Nu')</p> <p>c) if eoln(f)=false then write('Final de fișier') else write('Nu')</p> <p>d) if not eof(f) then write('Nu') else write('Final de fișier')</p> | <p>a) if (eof(f)) cout &lt;&lt; "Final de fișier"; else cout &lt;&lt; "Nu";</p> <p>b) if (feof(f)) cout &lt;&lt; "Final de fișier"; else cout &lt;&lt; "Nu";</p> <p>c) if (!eof(f)) cout &lt;&lt; "Final de fișier"; else cout &lt;&lt; "Nu";</p> <p>d) if (!feof(f)) cout &lt;&lt; "Nu"; else cout &lt;&lt; "Final de fișier";</p> |
|---|---|

5. Știind că fișierul 'D.TXT' are următorul conținut, ce se va afișa în urma executării programului următor?

D.TXT

```
13 45 23
32 42 234
56 78
32 23 43
32 32 32 32
```

- |   |   |
|---|---|
| <pre>var f:text; x,y,z:integer; begin   assign(f, 'D.txt'); reset(f);   readln(f,x,y);   read(f,z);   writeln(x, ' ', y, ' ', z, ' '); end.</pre> | <pre>#include &lt;stdio.h&gt; FILE *f; int x,y,z; void main() {   f=fopen("D.txt", "r");   fscanf(f, "%d %d %d\n", &amp;x, &amp;y, &amp;z);   fscanf(f, "%d", &amp;z);   printf("%d %d %d\n", x, y, z); }</pre> |
| <p>a) 13 45 23                      b) 13 32 56</p>   | <p>c) 13 32 23                      d) 13 45 32</p>   |

6. Știind că fișierul 'E.TXT' are următorul conținut, ce se va afișa în urma executării programului următor?

E.TXT

```
1 2 3 4
67 34 23
567 546 677
1234 3234 4565 6564
12345 12445 12223
```

- |   |  |
|---|--|
| <pre>var   f:text; x,y,z:integer; begin   assign(f, 'E.txt'); reset(f);   read(f,x,y);   readln(f,z);   writeln(x, ' ', y, ' ', z, ' '); end.</pre> | <pre>#include &lt;stdio.h&gt; FILE *f; int x,y,z; void main() {   f=fopen("E.txt", "r");   fscanf(f, "%d %d", &amp;x, &amp;y);   fscanf(f, "%d\n", &amp;z);   printf("%d %d %d\n", x, y, z); }</pre> |
| <p>a) 1 2 3                      b) 1 2 67                      c) 1 67 567                      d) 2 3 4</p>                                       |  |

7. Știind că fișierul 'F.TXT' are următorul conținut, ce se va afișa în urma executării programului următor?

F.TXT

```
12345 12
93 16 32
8.023 322 21
0.823 21
0.21 213.12
```

- |   |   |
|---|---|
| <pre>var   f:text;   x,y,z:char; begin   assign(f, 'F.txt'); reset(f);   read(f,x); readln(f,y);   read(f,z);   writeln(x, ' ', y, ' ', z, ' '); end.</pre> | <pre>#include &lt;stdio.h&gt; FILE *f; char x,y,z; void main() {   f=fopen("F.txt", "r");   fscanf(f, "%c", &amp;x);   fscanf(f, "%c", &amp;y);   while(getc(f) != '\n');   fscanf(f, "%c", &amp;z);   printf("%c %c %c\n", x, y, z); }</pre> |
| <p>a) 1 2 3                      b) 1 2 9                      c) 1 9 8                      d) 1 9 3</p>   |   |

8. Știind că fișierul 'G.TXT' are următorul conținut, ce se va afișa în urma executării programului următor?

G.TXT

```
IaEaws
S,a,d sdll
-dsa
Xda dsa asd
da
sda sad
```

```

var
  f:text;
  x,y,z:char;
begin
  assign(f,'G.txt'); reset(f);
  readln(f);
  readln(f,x);
  readln(f,y);
  readln(f,z);
  writeln(x,' ',y,' ',z,' ');
end.

#include <stdio.h>
FILE *f; char x,y,z;
void main() {
  f=fopen("G.txt","r");
  while(getc(f)!='\n');
  fscanf(f,"%c",&x);
  while(getc(f)!='\n');
  fscanf(f,"%c",&y);
  while(getc(f)!='\n');
  fscanf(f,"%c",&z);
  printf("%c %c %c\n",x,y,z);
}

```

a) I S -                      b) a S -                      c) S - X                      d) I E S

9. Care va fi conținutul fișierului *H.TXT* în urma executării programului următor?

```

var
  f:text;
  x,y,z:char;
begin
  assign(f,'H.txt'); rewrite(f);
  writeln(f,'Ieri ', 13);
  write('Azi ', 14);
  writeln('Maine ',15);
  close(f);
end.

#include <stdio.h>
FILE *f; char x,y,z;
void main() {
  f=fopen("H.txt","w");
  fprintf(f,"Ieri %d\n", 13);
  fprintf(f,"Azi %d ", 14);
  fprintf(f,"Maine %d\n", 15);
  fclose(f);
}

```

a) Ieri  
13 Azi 14  
Maine  
15

b) Ieri 13  
Azi 14 Maine 15

c) Ieri 13  
Azi 14  
Maine 15

d) Ieri  
13 Azi 14 Maine  
15

### 2.3.2 Probleme rezolvate

1. Fișierul text *IN.TXT* suferă următoarea prelucrare: de pe fiecare linie sunt șterse toate caracterele cu excepția primului și ultimului. Realizați un program care efectuează această operație asupra conținutului fișierului *IN.TXT*.

**Soluție:** Se va folosi un fișier auxiliar *O.TXT* pentru scrierea pe fiecare linie a primului și ultimului caracter de pe linia corespunzătoare din *IN.TXT*. După această operație fișierul *IN.TXT* va fi șters, iar *O.TXT* va fi redenumit cu numele *IN.TXT*.

```

1 var f,g:text;
2   ch:char;
3   ok:boolean;
4 begin
5   assign(f,'in.txt'); reset(f);
6   assign(g,'o.txt'); rewrite(g);

#include <stdio.h>
#include <string.h>
char s[256];
void main() {
  FILE *f1=fopen("in.txt","r");
  FILE *f2=fopen("o.txt","w");
}

```

```

7 while not(eof(f)) do begin
8   read(f,ch);
9   write(g,ch);
10  ok:=true;
11  while not(eoln(f)) do begin
12    read(f,(ch));
13    ok:=false;
14  end;
15  if not ok then write(g,ch);
16  readln(f); writeln(g);
17 end;
18 close(f); close(g);
19 erase(f); rename(g,'in.txt');
20 end.

while (!feof(f1)) {
  fscanf(f1,"%s",&s);
  if (strlen(s)==1)
    fprintf(f2,"%c\n",s[0]);
  else {
    fprintf(f2,"%c",s[0]);
    fprintf(f2,"%c",s[strlen(s)-1]);
    fprintf(f2,"\n");
  }
}
fclose(f1); fclose(f2);
unlink("in.txt");
rename("o.txt","in.txt");
}

```

2. Un fișier text conține numere întregi dispuse pe mai multe linii. Se dorește înlocuirea în fișier a tuturor aparițiilor unui număr *x* cu un alt număr *y*, citit de la tastatură. Realizați un program care permite efectuarea acestei modificări asupra conținutului fișierului *IN.TXT*.

**Soluție:** Se folosește un fișier auxiliar *OUT.TXT* deschis la operația de scrierea, în care se vor plasa toate numerele din *IN.TXT*, excepție făcând *x* care va fi înlocuit cu *y*. După această operație fișierul *IN.TXT* va fi șters iar *OUT.TXT* va fi redenumit cu numele *IN.TXT*.

```

1 var f,g:text;
2   x,y,nr:integer;
3 begin
4   assign(f,'in.txt'); reset(f);
5   assign(g,'out.txt'); rewrite(g);
6   writeln('nr de cautat:');
7   read(x);
8   writeln('Inlocuiesc cu');
9   read(y);
10  while not(eof(f)) do begin
11    while not eoln(f) do begin
12      read(f,nr);
13      if nr=x then write(g,y,' ');
14      else write(g,nr,' ');
15    end;
16    readln(f); writeln(g);
17  end;
18  close(f); close(g);
19  erase(f);
20  rename(g,'in.txt');
21 end.

#include <stdio.h>
int x,y,numar; char c;
void main() {
  printf("nr de cautat:");
  scanf("%d",&x);
  printf("Inlocuiesc cu:");
  scanf("%d",&y);
  FILE *f1=fopen("in.txt","r");
  FILE *f2=fopen("out.txt","w");
  while (!feof(f1)) {
    fscanf(f1,"%d%c",&numar,&c);
    if (x==numar)
      fprintf(f2,"%d ",y);
    else
      fprintf(f2,"%d ",numar);
    if (c=='\n')
      fprintf(f2,"\n");
  }
  fclose(f1); fclose(f2);
  unlink("in.txt");
  rename("out.txt","in.txt");
}

```

3. În fișierul text *IN.TXT* există dispuse pe fiecare linie câte un număr din șirul numerelor primelor *n* ( $n < 30000$ ) numere naturale nenule. Excepție face un singur număr care a fost omis. Realizați un program care determină numărul lipsă.

Exemplu:

IN.TXT

Se va afișa: 3

2  
4  
5  
1

**Soluție:** Suma primelor  $n$  numere naturale este  $n*(n+1)/2$ . Se va calcula suma numerelor din fișier și numărul acestora. Pe baza formulei de mai sus se determină numărul lipsă.

```
1 var f,g:text; #include <stdio.h>
2 aux,suma,nr:longint; int nr,aux; long suma;
3 begin void main() {
4 assign(f,'in.txt'); reset(f); FILE *f=fopen("in.txt","r");
5 nr:=0; while (!feof(f)) {
6 while not(eof(f)) do begin fscanf(f,"%d\n",&aux);
7 readln(f,aux); suma:=(long)aux;
8 suma:=suma+aux; nr++;
9 inc(nr); }
10 end; fclose(f);
11 close(f); nr:=(long)(nr+1)*(nr+2)/2-
12 nr:=(nr+1)*(nr+2) div 2-suma; (long)suma;
13 writeln('Numarul lipsa=',nr); printf("Nr.lipsa=%d\n",nr);
14 end. }
```

4. În fișierul *IN.TXT* există pe prima linie un șir crescător de numere naturale. Citirea din fișier a unui nou număr este condiționată de obținerea, ca sumă formată din termeni distincți citiți din fișier, a unui șir de numere consecutive începând cu 1. Să se determine care este numărul maxim care se poate obține respectând regula dată.

Exemplu: Pentru fișierul:

IN.TXT

1 2 4 10 11 32 324 1 54321

Se va afișa 7

Citirea se va încheia o dată cu citirea valorii 10, deoarece valoarea 8 nu se poate forma ca sumă de termeni preluați din fișier.

1, 2, 3(1+2), 4, 5(1+4), 6(2+4), 7(1+2+4)

**Soluție:** Se citesc succesiv numere din fișier, până la întâlnirea unui număr care depășește cu mai mult de o unitate suma numerelor anterior citite. Fie  $S$  această sumă. În această situație valoarea  $S+1$  nu se mai poate obține.

```
1 var f:text; #include <stdio.h>
2 ok:boolean; int n,curent;
3 suma,curent:longint; long suma;
```

```
4 begin
5 assign(f,'in.txt'); reset(f);
6 suma:=0;
7 ok:=true;
8 while not(eof(f)) and ok do
9 begin
10 read(f,curent);
11 if curent>suma+1 then
12 ok:=false
13 else suma:=suma+curent;
14 end;
15 close(f);
16 writeln('Nr. maxim ',suma);
17 end.

void main() {
int i;
FILE *f=fopen("in.txt","r");
do {
fscanf(f,"%d",&curent);
if (curent>suma+1) {
printf("Nr. maxim ");
printf("%d\n",suma);
return;
}
suma+=curent;
} while (!feof(f));
}
```

5. Să se înlocuiască toate caracterele neimprimabile din fișierul text *A.IN* prin codul ASCII al lor, precedat de caracterul „#”.

**Soluție:** Un caracter neimprimabil are codul ASCII mai mic sau egal cu 2. Fișierul *A.IN* va fi parcurs la citire și simultan se va crea fișierul *B.IN* în care caracterele vor fi scrise conform cerinței. La final, fișierul inițial va fi șters, iar *B.IN* va fi redenumit cu numele *A.IN*.

```
1 var f,g,h:text; #include <stdio.h>
2 m,i,j:integer; FILE *f,*g,*h;
3 x:char; int m,i,j; char x;
4 begin void main() {
5 assign(f,'a.in'); f=fopen("a.in","r");
6 reset(f); g=fopen("b.in","w");
7 assign(g,'b.in'); while (!feof(f)) {
8 rewrite(g); do {
9 while not eof(f) do begin fscanf(f,"%c",&x);
10 while not eoln(f) do begin if (x=='\n' || feof(f)) break;
11 read(f,x); if (x<=32)
12 if ord(x)<=32 then fprintf(g,"%d", (int)x);
13 write(g,'#',ord(x)); else fprintf(g,"%c",x);
14 else write(g,x); } while (1);
15 end; fscanf(f,"%n");
16 writeln(g); readln(f); fprintf(g,"%n");
17 end; }
18 close(f); close(g); fclose(f); fclose(g);
19 erase(f); unlink("a.in");
20 rename(g,'a.in') rename("b.in","a.in");
21 end. }
```

6. Fie fișierul text *IN.TXT* ce conține doar caractere alfanumerice. Realizați un program care creează fișierul *OUT.TXT* în care se regăsesc caracterele situate pe poziții pare (al doilea, al patrulea, ș.a.m.d.) din cadrul liniilor cu număr de ordine impar.

Exemplu: *IN.TXT*

ADFABET

M23CRI

123456789

*OUT.TXT*

DAE

2468

**Soluție:** La parcurgerea fișierului *A.IN* vor fi ignorate la operația de scriere liniile cu număr de ordine par și caracterele situate pe poziții impare în cadrul acestora.

```
1 var f,g,h:text; x:char;
2 begin
3   assign(f,'a.in'); reset(f);
4   assign(g,'b.in'); rewrite(g);
5   while not eof(f) do begin
6     while not eoln(f) do begin
7       read(f,x);
8       if not eoln(f) then begin
9         read(f,x); write(g,x)
10      end;
11    end;
12    readln(f);
13    readln(f);
14    writeln(g);
15  end;
16  close(f);
17  close(g);
18 end.
```

```
#include <stdio.h>
FILE *f,*g,*h; char x;
void main() {
  f=fopen("in.txt","r");
  g=fopen("out.txt","w");
  while (!feof(f)) {
    do
    {
      fscanf(f,"%c",&x);
      if (x=='\n' || feof(f)) break;
      fscanf(f,"%c",&x);
      if (x=='\n' || feof(f)) break;
      fprintf(g,"%c",x);
    } while (1);
    while (!feof(f) && getc(f)!='\n');
    fprintf(g,"\n");
  }
  fclose(f); fclose(g);
}
```

7. Scrieți un program care verifică dacă două fișiere text *I1.TXT* și *I2.TXT* au conținut identic.

**Soluție:** Pentru a verifica dacă două fișiere au conținut identic vor trebui parcurse simultan la citire și verificată egalitatea caracter cu caracter.

Nu trebuie scăpat din vedere că o linie a unui fișier poate reprezenta prefixul liniei corespunzătoare celui alt fișier. De exemplu, pe o linie se pot găsi caracterele 'eu si mama', iar pe linia corespunzătoare din celălalt fișier 'eu si mama vin'.

```
1 var f,g:text;
2   ok:boolean;
3   x,y:char;
4 begin
5   assign(f,'i1.txt'); reset(f);
6   assign(g,'i2.txt'); reset(g);
7   ok:=true;
8   while not eof(f) and not eof(g)
9   do begin
10    while not eoln(f) and not eoln(g)
11    do begin
12      read(f,x); read(g,y);
13      if x<y then ok:=false;
14    end;
15    if eoln(f)<>eoln(g) then
16      ok:=false;
17    readln(f); readln(g);
18  end;
19  if eof(f)<>eof(g) then
20    ok:=false;
```

```
#include <stdio.h>
FILE *f,*g;
int ok;
char x,y;
void main() {
  f=fopen("i1.txt","r");
  g=fopen("i2.txt","r");
  ok=1;
  while (!feof(f) && !feof(g)) {
    do {
      fscanf(f,"%c",&x);
      fscanf(g,"%c",&y);
      if (x=='\n' || y=='\n' ||
          feof(f) || feof(g)) break;
      if (x!=y) ok=0;
    } while (1);
    if (x!=y) ok=0;
    fscanf(f,"%c",&x); fscanf(g,"%c",&y);
  }
  if (feof(f)!=feof(g)) ok=0;
```

```
21 if ok then
22   writeln('Continut identic')
23 else
24   writeln('Continut diferit')
end.
```

```
if (ok)
  printf("Continut identic\n");
else
  printf("Continut diferit\n");
}
```

8. Fișierul text *IN.TXT* conține pe prima linie un șir de caractere alfanumerice. Creați un alt fișier text *OUT.TXT* în care să se regăsească conținutul din *IN.TXT* dispus pe mai multe linii în felul următor: pe prima linie primul caracter, pe a doua linie următoarele două caractere ș.a.m.d, până când au fost plasate toate caracterele. Ultima linie a fișierului *OUT.TXT* poate avea mai puține caractere decât linia precedentă.

<b>Exemplu:</b>	<i>IN.TXT</i>	<i>OUT.TXT</i>
	MWQDWEFR	M
		WQ
		DWE
		FR

**Soluție:** Scrierea în fișierul *OUT.TXT* se va realiza simultan cu citirea caracterelor din *IN.TXT*. Pentru respectarea formatului impus la scriere, ne vom folosi de o variabilă care contorizează numărul liniei curente pe care se face scrierea.

```
1 var f,g:text; x:char;
2   ln,i,j:integer;
3 begin
4   assign(f,'in.txt'); reset(f);
5   assign(g,'out.txt'); rewrite(g);
6   ln:=0;
7   while not eof(f) do begin
8     inc(ln); j:=0;
9     while (j<ln) and not eof(f) do
10      begin
11        read(f,x);
12        write(g,x);
13        inc(j);
14      end;
15      writeln(g);
16    end;
17    close(f);
18    close(g);
19 end.
```

```
#include <stdio.h>
FILE *f,*g; char x;
int ln,i,j;
void main() {
  f=fopen("in.txt","r");
  g=fopen("out.txt","w");
  ln=0;
  while (!feof(f)) {
    ln++; j=0;
    while (j<ln && !feof(f)) {
      fscanf(f,"%c",&x);
      fprintf(g,"%c",x);
      j++;
    }
    fprintf(g,"\n");
  }
  fclose(f);
  fclose(g);
}
```

9. În fișierul *TEXT.TXT* se află mai multe „parole” formate din caractere de tip majusculă, scrise fiecare pe câte o linie. Fiecareia i se asociază un număr obținut ca produs al numerelor de ordine ale literelor în alfabet. De exemplu pentru cuvântul „BAC” numărul asociat este  $2*1*3=6$ . Să se creeze fișierul *OUT.TXT* în care pe câte o linie se află parolele care au numărul asociat cel mai mare.

**Soluție:** La prima traversare a fișierului *TEXT.TXT* se va crea fișierul *P.TXT* în care sunt depuse numerele asociate fiecărei parole, câte unul pe fiecare linie. Ambele fișiere vor fi închise și apoi redeschise la operația de citire pentru crearea lui *OUT.TXT*.

```

1 var f,g,h:text; m,i,j:integer;
2   x:char;
3 begin
4   assign(f,'text.txt');
5   reset(f);
6   assign(g,'out.txt');
7   rewrite(g);
8   assign(h,'p.txt');
9   rewrite(h);
10  m:=0;
11  while not eof(f) do begin
12    j:=1;
13    while not eoln(f) do begin
14      read(f,x);
15      j:=j*(ord(x)-64);
16    end;
17    writeln(h,j); readln(f);
18    if j>m then m:=j;
19  end;
20  close(f); close(h);
21  reset(f); reset(h);
22  while not eof(f) do begin
23    readln(h,j);
24    if j=m then begin
25      while not eoln(f) do begin
26        read(f,x);
27        write(g,x);
28      end;
29      writeln(g);
30    end;
31    readln(f);
32  end;
33  close(f); close(g); close(h);
34 end.

```

```

#include <stdio.h>
FILE *f,*g,*h;
int m,i,j; char x;
void main() {
  f=fopen("text.txt","r");
  g=fopen("out.txt","w");
  h=fopen("p.txt","w"); m=0;
  while (!feof(f)) {
    j=1;
    do {
      fscanf(f,"%c",&x);
      if (x=='\n' || feof(f)) break;
      j*=(int)x-64;
    } while (1);
    fprintf(h,"%d\n",j);
    fscanf(f,"%n",&j);
    if (j>m) m=j;
  }
  fclose(f); fclose(h);
  f=fopen("text.txt","r");
  h=fopen("p.txt","r");
  while (!feof(f)) {
    fscanf(h,"%d\n",&j);
    if (j==m) {
      do {
        fscanf(f,"%c",&x);
        if (x=='\n' || feof(f)) break;
        fprintf(g,"%c",x);
      } while (1);
      fprintf(g,"%n");
    } else
      while (getc(f)!='\n' && !feof(f));
  }
  fclose(f); fclose(g); fclose(h);
}

```

10. Se consideră fișierul *text IN.TXT* ce conține numere întregi dispuse pe mai multe linii. Numerele sunt separate în cadrul liniilor prin caracterul virgulă “,”. Scrieți un program care creează un fișier *OUT.TXT* ce conține pe fiecare linie suma numerelor situate pe aceeași linie în fișierul *IN.TXT*.

**Soluție:** În problema de față separatorii unor date numerice sunt reprezentați de caracterul virgulă. Această situație impune folosirea la citire a unei variabile de tip *char*.

Există evident problema conversiei unei valori de tip caracter în valoare numerică și pe de altă parte construirea numărului citit cifră cu cifră.

Conversia unui caracter numeric în valoare numerică se poate face prin intermediul codului ASCII.

```

1 var f,g:text; x:char;
2   s,nr:integer;
3
4 begin
5   assign(f,'in.txt'); reset(f);
6   assign(g,'out.txt'); rewrite(g);
7   while not eof(f) do begin
8     s:=0; nr:=0;
9     while not eoln(f) do begin
10      read(f,x);
11      if x<>',' then
12        nr:=nr*10+ord(x)-48
13      else begin
14        s:=s+nr; nr:=0;
15      end;
16    end;
17    s:=s+nr; writeln(g,s);
18    readln(f);
19  end;
20  close(f); close(g);
21 end.

```

```

#include <stdio.h>
FILE *f,*g; char x;
int s,nr;
void main() {
  f=fopen("in.txt","r");
  g=fopen("out.txt","w");
  while (!feof(f)) {
    s=0; nr=0;
    do {
      fscanf(f,"%c",&x);
      if (x=='\n') break;
      if (x!=',' ) nr=nr*10+(int)x-48;
      else {
        s+=nr; nr=0;
      }
    } while (1);
    s+=nr;
    fprintf(g,"%d",s);
    fscanf(f,"%n",&j);
  }
  fclose(f); fclose(g);
}

```

### 2.3.3 Probleme propuse

1. Se consideră fișierul *IN.TXT* ce conține pe prima linie un număr *n* natural, iar pe a doua linie *n* numere întregi. Afișați pe ecran primul și ultimul număr de pe linia a doua.
2. Se consideră două fișiere *I1.TXT* și *I2.TXT*. Verificați care dintre ele conține mai multe caractere. Afișați pentru aceasta numele fișierului.
3. Se consideră fișierul *IN.TXT* care conține 10 numere întregi scrise fiecare pe câte o linie. Afișați prima și ultima cifră a fiecărui număr fără a prelua valorile într-un vector.
4. Se consideră fișierul *REAL.TXT* ce conține numere reale dispuse pe mai multe linii. Să se creeze un alt fișier *INTREG.TXT* în care să se regăsească valorile din primul fișier rotunjite fiecare dintre ele la cel mai apropiat întreg și dispuse în aceeași ordine.

Exemplu: *REAL.TXT*

```

2.3 4.05
1.0 12.8 3.45
1.93

```

*INTREG.TXT*

```

2 4
1 13 3
2

```

5. Se consideră fișierul *REAL.TXT* ce conține numere reale dispuse pe mai multe linii. Să se creeze un alt fișier *FRAC.TXT* în care să se regăsească părțile fracționare ale fiecărei valori din primul fișier, cu două zecimale exacte dar dispuse pe o singură linie.

Exemplu: *REAL.TXT*

2.3 4.05  
1.0 12.8 3.45  
1.93

*FRAC.TXT*

0.30 0.05 0.00 0.80 0.45 0.93

6. Se consideră două fișiere *I1.TXT* și *I2.TXT*. Unul conține numere reprezentând vârsta unor elevi, iar liniile corespunzătoare din celălalt fișier numele acestora, codificate printr-o majusculă. Să se afișeze pe ecran codificările numelor celor mai tineri elevi.

Exemplu: *I1.TXT*

13  
15  
16  
13

I  
M  
V  
D

*I2.TXT*

Se va afișa: I D

7. Considerăm fișierul *IN.TXT* care cuprinde pe fiecare linie caractere alfanumerice (litere și cifre). Creați un fișier cu numele *OUT.TXT* în care se regăsesc liniile din *IN.TXT* din care au fost eliminate cifrele.

8. Scrieți un program care crează un fișier text *I3.TXT* prin concatenarea conținuturilor a două fișiere text numite *I1.TXT* și *I2.TXT*.

9. Se consideră fișierul text *INPUT.TXT*. Să se scrie un program care crează un fișier *OUTPUT.TXT* ce conține liniile cu număr de ordine impar din *INPUT.TXT*.

10. Să se scrie un program care crează un fișier text *OUT.TXT* ce va conține pe o singură linie, codurile ASCII ale tuturor caracterelor ce se află în fișierul *IN.TXT*.

11. Să se scrie un program care determină caracterul neimprimabil (cod ASCII mai mic decât 32) cu frecvență de apariție maximă în fișierul *IN.TXT*.

12. Creați un program care transformă toate literele mici din fișierul *IN.TXT* în majuscule.

13. Avem în directorul curent fișierul text *IN.TXT* care conține caracterele alfanumerice. Considerăm că literele sunt separatorii numerelor. De exemplu, dacă pe o linie apar caracterele A23sc345ss5e, atunci ea conține trei numere separate prin câte un spațiu: 23 345 5. Realizați un program care crează fișierul *OUT.TXT* în care se regăsesc date de tip întregi preluate în ordine de pe liniile fișierului.

14. Se consideră două fișiere *I1.TXT* și *I2.TXT*. Unul conține pe fiecare linie câte două numere reprezentând notele la matematică ale unor elevi, iar liniile corespunzătoare din celălalt fișier numele acestora codificat printr-o majusculă. Să se creeze un nou fișier *I3.TXT* în care pe fiecare linie să se regăsească numele elevului și media la matematică exprimată cu două zecimale.

Exemplu: *I1.TXT*

10 8  
7 8  
5 9  
10 10  
6 8  
5 6

*I2.TXT*

I  
M  
V  
M  
U  
D

*I3.TXT*

I 9.00  
M 7.50  
V 7.00  
M 10.00  
U 7.00  
D 5.50

## 2.4 Probleme de concurs ce procesează date structurate

### 2.4.1 Probleme rezolvate

1. (Secvența de sumă maximă - \*\*\*). Să se afișeze secvența de sumă maximă dintr-un șir de numere întregi și valoarea acestei sume.

Exemplu: pentru  $n=8$  și șirul 2 -4 -3 5 -4 7 8 -2 se va afișa

suma=16

5 -4 7 8

**Soluție:** Vom opta pentru o rezolvare liniară a problemei, algoritmi de complexitate cubică și respectiv pătratică fiind neinteresanti în condiții de concurs.

La fiecare pas al parcurgerii vectorului a se impun efectuate următoarele operații:

1. verificarea semnului sumei secvenței curente (sc)

1.1 dacă  $sc > 0$  atunci la ea se va adăuga și elementul curent

1.2 dacă  $sc$  este negativă atunci secvența curentă se resetează, efectuându-se reinițializările sumei curente cu elementul  $a[i]$  ( $sc \leftarrow a[i]$ ) și a poziției de început  $pc$  cu  $i$  ( $pc \leftarrow i$ ).

2. actualizarea secvenței de sumă maximă dacă este posibil, cu păstrarea indicelui de început  $ic$  și a indicelui de sfârșit  $sf$ .

```
1 max ← a[1]; sc ← a[1]; ic ← 1; sf ← 1; pc ← 1;
2 pentru i ← 2, n executa
3   dacă sc > 0 atunci sc ← sc + a[i]
4   altfel
5     sc ← a[i];
6     pc ← i;
7   sf ← i;
```

2.3  
3 1  
5 1

4. (**Clasament - \*\*\*\***).  $N$  sportivi numerotați cu numere de la 1 la  $N$  iau parte la un maraton. Clasamentul final este codificat sub forma unui vector  $A$  de lungime  $N$ . Fiecare element  $a[i]$  din vector are următoarea interpretare: concurentul clasat pe locul  $i$  a devansat un număr de  $a[i]$  concurenți ale căror numere de pe tricou sunt mai mari decât al lui. În decursul ultimului an, toți acești  $N$  sportivi, au participat la  $M$  probe de maraton și de fiecare dată au avut același număr pe tricoul de concurs. Toate clasamentele finale au fost codificate după regula descrisă. Știind ca toți sportivii au terminat fiecare din cele  $M$  probe de maraton, aflați care concurenți au evoluat din ce în ce mai bine, adică la fiecare nouă probă locul pe care l-au ocupat a fost strict mai mic decât la proba anterioară.



În fișierul text *IN.IN* pe prima linie se află două numere  $N$  și  $M$ , despărțite printr-un spațiu. Pe următoarele  $M$  linii, în ordine cronologică a momentului desfășurării, clasamentele finale. În cadrul liniilor numerele sunt despărțite prin câte un spațiu. ( $0 < N < 3000$ ;  $0 < M < 10$ )

În fișierul text *OUT.OUT*, pe o singură linie, se vor scrie în ordine crescătoare, numerele de pe tricou ale concurenților identificați cu evoluții ascendente. În cadrul liniilor, numerele vor fi despărțite prin câte un spațiu. Dacă nu există soluție, fișierul de ieșire va conține valoarea 0.

IN.IN	OUT.OUT
5 2	1 4
3 2 2 1 0	
3 3 1 1 0	

Pentru prima probă de maraton clasamentul final a fost 2, 3, 1, 4, 5, iar pentru a doua probă 2, 1, 4, 3, 5. Concurentul cu tricoul 1 s-a clasat pe locul 3, iar la a doua probă pe locul 2. Concurentul cu tricoul 4 s-a clasat în ordine inițial pe locul 4 și apoi 3.

**Soluție:** În procesul de identificare a clasamentului final se începe cu elementul  $a[1]$  care indică numărul tricoului primului clasat:  $n - a[1]$  selectându-se acest număr. Se continuă identificarea ordinii la sosire ignorându-se numerele de tricou deja selectate. La finalul fiecărei probe, se reține într-un vector poziția în clasament a fiecărui concurent, marcându-se cei care nu au avut o evoluție ascendentă.

Ca structuri de date vom folosi trei tablouri unidimensionale :

- $A(N)$  reține clasamentul pentru fiecare probă
- $P(N)$  reține pentru fiecare concurent poziția ocupată în cadrul ultimei probe desfășurate. Dacă la proba curentă, concurentul  $j$ , nu a avut o evoluție ascendentă față de proba anterioară, atunci  $p[j] = n+1$
- $SEL(N)$  indică, pentru fiecare probă în parte, dacă un concurent a trecut sau nu linia de sosire.

```

1  pentru k = 1, m executa
2      pentru i = 1, n executa sel[i] ← False
3      pentru i = 1, n executa citește a[i];
4      pentru i = 1, n executa
5          j ← n + 1; nr ← 0;
6          cat timp (j > 1) and (nr ≤ a[i]) executa
7              j ← j - 1;
8              dacă not sel[j] atunci nr ← nr + 1;
9              sel[j] ← True;
10             dacă p[j] = 0 atunci p[j] ← i
11             altfel
12                 dacă (p[j] > i) and (p[j] ≠ n + 1) atunci p[j] ← i
13                 altfel p[j] ← n + 1
14             sfârșit
15         sfârșit
16     sfârșit

```

```

19  ┌─┐
20  │  │
21  │  │

```

5. (**Cerc - \*\*\***) Considerăm că  $2 \cdot n$  copii au tricouri numerotate cu numere de la 1 la  $2 \cdot n$ . Așezați copiii în cerc astfel încât extrăgându-i din  $k$  în  $k$  să părăsească cercul cei cu numere pare pe tricou, în ordine crescătoare, apoi cei cu numerele impare pe tricou, de asemenea în ordine crescătoare. Afișarea va începe de la primul copil care va fi extras din cerc.

**Exemplu:** pentru  $n=4$  și  $k=3$  copiii vor fi dispuși în cerc astfel: 2 5 8 4 7 3 6 1. Extragerea din 3 în 3, începând cu primul din listă se va face în ordinea cerută, adică: 2 4 6 8 1 3 5 7

**Soluție:** Algoritmul presupune completarea elementelor unui vector din  $k$  în  $k$  poziții ignorându-le pe cele deja "ocupate". Indicele curent  $x$  care va fi completat traversează prin incrementare toate cele  $2 \cdot n$  poziții, revenind la valoarea 1 după atingerea poziției  $2 \cdot n$ .

Vectorul  $a$  se completează inițial cu valorile pare 2, 4... $2 \cdot n$ , apoi cu cele impare 1, 3,... $2 \cdot n - 1$ .

```

1  x ← 1
2  pentru i = 1, 2 * n executa
3      nr ← 0;
4      cat timp nr < k executa sel[i] ← False
5          dacă x < 2 * n atunci x ← x + 1
6          altfel x ← 1
7          dacă a[x] = 0 atunci nr ← nr + 1
8          sfârșit
9      sfârșit
10     sfârșit
11     dacă i = 0 atunci x ← 1
12     sfârșit
13     dacă i ≤ n atunci a[x] ← 2 * i
14     altfel a[x] ← ((i - 1) mod n) * 2 + 1
15     sfârșit
16 sfârșit

```

6. (**Multiplu - \*\*\*\***) Fie  $n$  un număr întreg. Găsiți un număr în baza 10 divizibil cu  $n$  și format numai din cifre de 1 și 0. ( $n \leq 1000$ )

**Soluție:** Vom considera șirul numerele 1, 11, 111, ..., 11...1 (ultimul având  $n$  cifre de 1). Putem avea două situații:

- Printre numere există unul care este multiplu de  $n$ . Rezultă că acesta este numărul căutat.
- Toate numerele anterioare dau la împărțirea cu  $n$  resturi nenule, adică 1, 2, ...,  $n-1$ . Întrucât avem  $n$  numere și  $n-1$  resturi, conform principiului lui Dirichlet există două numere care dau același rest. Obținem astfel că diferența acestor două numere este numărul căutat (care este, evident, format din cifrele 0 și 1).

Numerele din șir nu pot fi memorate ca întregi (datorită numărului mare de cifre). De fapt ne interesează numai resturile lor la împărțirea cu  $n$ . Notăm cu  $x$  restul obținut pentru un număr oarecare din șir, format din  $i$  cifre de 1. Atunci restul numărului ce conține  $i+1$  cifre de 1 se obține ca  $(x*10+1) \bmod n$ .

Pentru implementare vom folosi un singur tablou unidimensional  $A(N-1)$ . Elementul  $a[r]$  va reprezenta numărul de cifre de 1 din care este format un număr din șir, care dă restul  $r$  la împărțirea la  $n$ .

```

1  a[1] ← 1; x ← 1;
2  nr ← 1; ok ← True;
3  cat timp ok executa
4    nr ← nr + 1;
5    r ← (x*10 + 1) mod n
6    daca r ≠ 0 atunci
7      daca a[r] = 0 atunci
8        a[r] ← nr;
9        x ← r
10     altfel
11       pentru i=1, nr - a[r] executa scrie 1
12       pentru i=1, a[r] executa scrie 0
13     ok ← False
14   altfel
15     ok ← False
16   pentru i = 1, nr executa scrie 1
17   ok ← False
18   pentru i = 1, nr executa scrie 1
19   ok ← False
20   pentru i = 1, nr executa scrie 1
21   ok ← False
22   pentru i = 1, nr executa scrie 1

```

7. (Adunare - \*\*) Se consideră două numere  $A$  și  $B$  cu cel mult 100 de cifre. Să se scrie un program care determină suma celor două numere.

Soluție: Se rețin numerele în vectori  $[0...100]$  astfel: elementul 0 va indica numărul de cifre și elementele de la 1 încolo vor reține cifrele în ordine inversă. De exemplu numărul 125 este reținut ca (3, 5, 2, 1).

Această modalitate de reprezentare ușurează simularea operației de adunare folosind algoritmul învățat la matematică.

```

1  t ← 0; i ← 1; //t este cifra de transport = rest
2  cat timp (i ≤ a[0]) or (i ≤ b[0]) or (t > 0) executa
3    t ← t + a[i] + b[i]; //adun cifre corespunzatoare + rest
4    a[i] ← t mod 10;
5    t ← t div 10; //noua valoare a cifrei de transport
6    i ← i + 1;
7  a[0] ← i - 1; //numarul de cifre al sumei

```

8. (Înmulțire - \*\*) Se consideră un număr  $A$  cu cel mult 100 de cifre și  $B$  un număr mai mic ca 32.768. Să se scrie un program care înmulțește cele două numere.

Soluție: Primul număr este reprezentat ca la operația de adunare. Se simulează operația de înmulțire folosind algoritmul învățat la matematică.

```

1  t ← 0; i ← 1; //t este cifra de transport = rest
2  cat timp (i ≤ a[0]) or (t > 0) executa
3    t ← t + a[i]*b;
4    a[i] ← t mod 10;
5    t ← t div 10; //noua valoare a cifrei de transport
6    i ← i + 1;
7  a[0] ← i - 1; //numarul de cifre al produsului

```

9. (Scădere - \*\*) Se consideră două numere  $A$  și  $B$ , cu cel mult 100 de cifre. Să se scrie un program care calculează diferența  $A-B$ .

Soluție: Numerele sunt reprezentate ca la operația de adunare. Se simulează operația de scădere folosind algoritmul învățat la matematică.

```

1  t ← 0; i ← 1;
2  cat timp (i ≤ a[0]) or (t > 0) executa
3    a[i] ← a[i] - b[i] - t;
4    daca a[i] < 0 atunci t ← 1; altfel t ← 0;
5    a[i] ← a[i] + t*10;
6    i ← i + 1;
7  cat timp (a[0] > 0) and (a[a[0]] = 0) executa
8    a[0] ← a[0] - 1;
9  a[0] ← a[0] - 1;

```

10. (Împărțire - \*\*) Se consideră un număr cu cel mult 100 de cifre și altul mai mic ca 32.768. Să se scrie un program care determină câtul împărțirii celor două numere.

Soluție: Primul număr este reprezentat ca la operația de adunare. Se simulează operația de împărțire folosind algoritmul învățat la matematică.

```

1  t ← 0; i ← a[0];
2  cat timp i > 0 executa
3    t ← t*10 + a[i];
4    a[i] ← t div b;
5    t ← t mod b;
6    i ← i-1;
7  cat timp (a[0] > 0) and (a[a[0]] = 0) executa
8    a[0] ← a[0] - 1;
9  a[0] ← a[0] - 1;

```

11. (Păianjen - \*\*\*) Să ne imaginăm o rețea formată din noduri situate în punctele de coordonate întregi, fiecare nod fiind unit prin bare paralele cu axele de coordonate de cele 4 noduri vecine. Un păianjen este plasat inițial în originea sistemului de coordonate. La fiecare secundă, păianjenul se poate deplasa din nodul în care se află în unul dintre cele 4 noduri vecine.

Scrieți un program care să determine în câte moduri se poate deplasa păianjenul din poziția inițială, într-o poziție finală dată, în timpul cel mai scurt.

Fișierul de intrare spider.in conține pe o singură linie abscisa și ordonata punctului final, separate prin spațiu:  $x\ y$  ( $0 < x, y \leq 80$ )

În fișierul de ieșire spider.out se va afișa pe prima linie numărul de moduri determinat Nr.

Exemplu:

2 3	spider.in	10	spider.out
	(Olimpiada Județeană de Informatică Gimnaziu, 2001, cls. VII-VIII)		

Soluție: Se calculează pentru fiecare poziție  $(x, y)$  în câte moduri se poate ajunge acolo astfel:  $A[x, y] = A[x-1, y] + A[x, y-1]$ .

```

1  pentru i ← 0, x executa
2    a[0][i] ← 1;
3
4  pentru i ← 0, y executa
5    a[i][0] ← 1;
6
7  pentru i ← 1, x executa
8    pentru j ← 1, y executa
9      a[i][j] ← a[i-1][j] + a[i][j-1];
10
11
12 scrie a[x][y];

```

12. (Șiruri - \*\*) Se dau două șiruri de lungime  $N$  și un număr  $K$  ( $N < 1000, K < 1000$ ). Cele două șiruri au numai numere 1 și -1. Scopul este să-l transformăm pe primul în al doilea. Singura operație permisă este să selectăm o secvență de  $K$  elemente alăturate și să le inversăm semnul la toate numerele cuprinse în această zonă. Secvența poate să înceapă cel mai devreme la primul element și să se sfârșească cel mai târziu la ultimul.

Fișierul de intrare siruri.in conține pe prima linie  $N$  și  $K$  separate printr-un spațiu. Pe următoarele  $N$  linii vor fi câte un număr (1 sau -1) reprezentând elementele primului șir (în ordinea dată), iar pe următoarele  $N$  linii cel de-al doilea șir. Fișierul de ieșire siruri.out va conține pe prima linie  $M$  numărul minim de operații, iar pe următoarele  $M$  linii poziția de început de unde se aplică operația (șirurile încep la poziția 1 și se termină în poziția  $N$ ). Se garantează că mereu există soluție.

Exemplu: siruri.in

```

4 2
1
-1
1
-1
-1
-1
1
1

```

siruri.out

```

3
1
2
3

```

Soluție: Se parcurg cele două șiruri poziție cu poziție, iar unde diferă se face o operație în poziția respectivă. La sfârșit se verifică dacă cele două șiruri sunt egale.

```

1  pentru i ← 1, n - k + 1 executa
2    daca a[i] ≠ b[i] atunci
3      pentru j ← i, i+k-1 executa
4        a[j] ← -a[j];
5
6      scrie i;
7
8

```

13. (Semne - \*\*\*\*) Pentru un număr  $N$  ( $0 < N < 1000$ ) natural nenul, să se găsească o combinație de semne + și -, adică un vector  $x = (x_1, x_2, \dots, x_k)$ ,  $x_i$  din mulțimea  $\{-1, 1\}$  astfel încât:  $N = x_1 \cdot 1^2 + x_2 \cdot 2^2 + \dots + x_k \cdot k^2$ , unde  $k$  este număr natural ce reprezintă numărul operatorilor folosiți.

Fișierul semne.in conține pe fiecare linie valorile lui  $n$  pentru care se doresc reprezentări ca mai sus.

Fișierul semne.out va conține pe câte o linie combinația de semne corespunzătoare fiecărui număr de pe aceeași linie din fișierul de intrare.

Exemplu:

semne.in

```

2
4
8
5

```

semne.out

```

---+
+---+
+---+---+
++---+

```

Soluție: Soluția se bazează pe inducție după  $n$  și pe următoarea observație:

$$x^2 - (x+1)^2 - (x+2)^2 + (x+3)^2 = 4.$$

Astfel, dacă avem o soluție pentru  $m$  putem construi o soluție pentru  $m+4$ . Soluția se bazează pe construirea secvenței de semne pentru 0, 1, 2 sau 3 (în funcție de restul lui  $n$  la 4) și adăugarea șirului  $+-+$  de  $n/4$  ori.

Pseudocodul este prezentat în continuare.

```

1  daca n mod 4 = 1 atunci
2    scrie "+";

```

```

3 altfel
4   daca n mod 4=2 atunci
5     scrie "----+";
6 altfel
7   daca n mod 4=3 atunci
8     scrie "--+";
9
10
11
12 pentru i=1,n div 4 executa
13   scrie "+---+";
14

```

14. (Text - \*\*) Dezamăgit de rezultatele sale la ultimul concurs, Paftenie a renunțat la programare și s-a concentrat strict asupra muncii laborioase, dar care implică mai puțin efort intelectual. De această dată, el primește un text de cel mult 1.000.000 de caractere, și trebuie să calculeze lungimea medie a cuvintelor textului, un cuvânt fiind definit ca o secvență continuă maximală de caractere ale alfabetului englez ('a' .. 'z', 'A' .. 'Z'). Definim lungimea medie = (lungimea totală a cuvintelor textului) / (numărul de cuvinte ale textului). Scrieți un program care îi rezolvă problema lui Paftenie.

Fișierul de intrare text.in conține textul dat.

Fișierul de ieșire text.out va conține pe prima linie un singur întreg, reprezentând partea întreagă a lungimii medii a cuvintelor textului.

Exemplu:

text.in		text.out
- Lasa-ma in pace, ca am invatat azi noapte toata ziua!	3	

(<http://infoarena.devnet.ro>)

Soluție:

Se parcurge fișierul caracter cu caracter (nu este necesară stocarea datelor de intrare într-un vector) și se mențin două variabile care indică poziția de început și sfârșit a ultimului cuvânt detectat până în prezent, dacă s-a găsit vreunul. De asemenea se păstrează și două variabile pentru suma lungimilor cuvintelor și numărul de cuvinte pentru a calcula rezultatul.

Atenție însă la sfârșitul parcurgerii fișierului de ieșire, dacă ultimul caracter citit a fost o literă mare sau mica, să se actualizeze numărul de cuvinte și suma lungimilor.

```

1 a ← -1;
2 b ← -1;
3 s ← 0;
4 nr ← 0;
5 cat timp nu s-a ajuns la sfârșitul fișierului executa
6   citește c;
7

```

```

8   daca c este litera mica sau mare atunci
9     daca a = -1 atunci
10      a ← 0; b ← 0;
11     altfel b ← b + 1;
12
13   altfel
14     daca a ≠ -1 atunci
15       s ← s + b - a + 1;
16       nr ← nr + 1;
17
18   a ← -1;
19   b ← -1;
20
21
22   daca a ≠ -1 atunci
23     s ← s + b - a + 1;
24     nr ← nr + 1;
25
26 scrie [s/nr]

```

15. (Palindrom cubic - \*\*) Se dă  $n$  număr natural. Găsiți cel mai mare număr cub perfect mai mic sau egal cu  $n$  care este și număr palindrom. Fișierul text cub.in conține pe prima linie numărul  $n$  ( $n < 2000000000$ ). Fișierul text cub.out va conține o linie pe care se află numărul cerut de problemă.

Exemplu:	cub.in		cub.out
	1340	1331	

Soluție: Se ridică la puterea a treia orice număr începând cu 1 până la  $\sqrt[n]{n}$ . Cubul obținut este verificat dacă este un număr palindrom.

```

1 i ← 1; nr ← 0;
2 cat timp i*i*i ≤ n executa
3   j ← i*i*i;
4   c ← 0;
5   cat timp j>0 executa
6     c ← c + 1;
7     a[c] ← j mod 10;
8     j ← j div 10;
9
10  ok ← true;
11  pentru j = 1, c div 2 executa
12    daca a[j] ≠ a[c-j+1] atunci ok ← false;
13
14  daca (ok=true) and (nr < i*i*i) atunci
15    nr ← i*i*i;
16
17  i ← i + 1;
18
19 scrie nr;

```

## 2.4.2 Probleme Propuse

1. (**Ultima cifra - \*\***) Se consideră un număr natural  $N$  mai mic decât  $10^5$ . Să se determine ultima cifră nenulă a factorialului ( $N! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot N$ )

Exemplu: Pentru  $N=5$  se va afișa: 2.

2. (**Cifre - \***) Se consideră un șir de  $n$  ( $n < 100$ ) numere întregi de cel mult 9 cifre. Să se verifice dacă:

- Fiecare număr din șir are cel puțin o cifră care apare și în numărul anterior.
- Fiecare număr din șir se obține din numărul anterior prin adăugarea sau eliminarea unei cifre.

Toate numerele conțin aceleași cifre. (Cifrele pot apărea de mai multe ori într-un număr și în orice ordine.). Pentru fiecare dintre cele 3 cerințe se va afișa pe ecran pe câte o linie unul din cuvintele DA respectiv NU reprezentând răspunsul corect la cerința precizată.

Exemplu:  $n=7$

173 17 7 72 472 4572 572  
a. DA b. DA c. NU

$n=4$   
177 17 117 1117  
a. DA b. DA c. DA

3. (**Exponent - \*\***) Se consideră un număr natural  $n$  ( $n \leq 1000$ ) și un număr natural  $p$ . Se cere să se afișeze exponentul maxim  $E$  astfel încât produsul  $1 \cdot 2 \cdot \dots \cdot n$  să fie divizibil cu  $p^E$ .

Exemplu: Pentru:  $n=7$  și  $p=6$  se va afișa 2. ( $1 \cdot 2 \cdot 3 \cdot \dots \cdot 7 = 5040$  și 5040 este divizibil cu 6, cu  $6^2=36$ , dar nu este divizibil cu  $6^3=216$ )

4. (**Piese de joc - \*\*\***) Se consideră o piesă de forma de mai jos. Toate pătrățelele piesei sunt colorate cu o aceeași culoare  $k$ . Se consideră o tablă de joc formată din  $n \times m$  pătrățele ( $n, m \leq 100$ ), fiecare fiind colorată cu o anumită culoare. Culoarele sunt codificate cu numere de la 1 la 100.



O astfel de piesă se poate așeza pe tablă dacă toate pătrățelele din tablă pe care le acoperă au aceeași culoare  $k$  și nici o altă pătrățică din jurul ei nu are culoarea  $k$  (în urma așezării piesei pe tablă). O pătrățică se află "în jurul" piesei dacă ea are cel puțin o latură comună cu aceasta.

Pentru  $n, m, k$  și o configurație a tablei date, să se determine numărul maxim de piese ce se pot așeza pe tablă. Fișierul `piese.in` conține pe prima linie numerele  $n, m$  și  $k$ , iar pe următoarele  $n$  linii câte  $m$  numere reprezentând codificarea tablei. Fișierul `piese.out` va conține numărul maxim determinat.

Exemplu: `piese.in`  
7 5 2  
2 2 5 4 5  
2 1 2 4 2  
2 3 2 2 2  
3 2 1 1 1  
3 2 1 2 2  
2 2 1 1 2  
3 4 1 1 2

`piese.out`  
3

(Sinaia Paco - 1997 clasa a VIII-a)

5. (**Intervale - \*\*\***) Se consideră un șir de  $N$  intervale de forma  $[A_i, B_i]$ , cu  $A_i, B_i$  numere întregi. Un interval poate fi eliminat din șirul celor  $N$  dacă există un alt interval care îl include strict pe acesta. Determinați numărul maxim de intervale care pot fi eliminate.

În fișierul text `interval.in` se găsește pe prima linie numărul  $N$  ( $N < 16000$ ), iar pe următoarele  $N$  linii perechi de numere naturale, mai mici decât 2000000000 ce reprezintă capetele intervalelor. Rezultatul va fi afișat pe ecran.

Exemplu: `interval.in`

5  
0 10  
2 9  
3 8  
1 15  
6 11

Se va afișa  
3

6. (**Coduri ascunse - \*\*\***) Pentru a deschide un seif trebuie introduse două parole. Ele sunt cuvinte formate din literele mari ale alfabetului englez. Seiful nu se deschide doar pentru o pereche de parole ci pentru oricare pereche care verifică următoarele reguli:

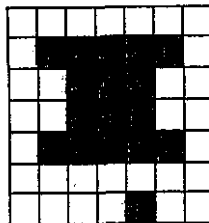
- conțin numai caracterele permise.
- Fiecare parolă are un număr asociat astfel: fiecare literă este reprezentată printr-un număr: A este 1, B este 2 și așa mai departe până la Z=26. Produsul numerelor asociate fiecărei litere din parolă reprezintă numărul asociat parolei. Exemplu: 'BAC' =  $2 \cdot 1 \cdot 3 = 6$ . Pentru ca o pereche de parole să fie acceptată trebuie ca numerele pe care le reprezintă fiecare să fie prime între ele.

Realizați un program care verifică dacă două cuvinte reprezintă parole corecte. Fișierul `parola.in` va conține pe câte o linie cele două cuvinte, iar `parola.out` pe prima linie numerele asociate celor două parole separate printr-un spațiu, iar pe a doua linie mesajul 'acceptat' sau 'neacceptat'

Exemplu: `parola.in`  
ABAC  
DA

`parola.out`  
6 7  
acceptat

7. (Aisberg - \*\*\*) Descriem un aisberg cu ajutorul unei matrici. Punctele marcate cu gri reprezintă pozițiile aparținând aisbergului. Dacă asupra lui suflă un vânt cald, el începe să se topească de pe margini spre interior. Regula topirii este următoarea: într-un interval de timp se topește acea porțiune de gheață care are cel puțin în două vecinătăți aer (notate cu 1). Astfel se produc alte astfel de câmpuri de aer (notate cu 2), care se vor topi în al doilea interval de timp ș.a.m.d. Scrieți un program care pentru un aisberg dat, returnează în câte intervale de timp se topește întreg aisbergul, respectiv pentru fiecare interval câte câmpuri de gheață mai are aisbergul.



Fișierul text 'aisberg.in' conține pe prima linie 2 numere întregi, separate printr-un spațiu, reprezentând numărul de linii ( $1 \leq N \leq 40$ ) și coloane ( $1 \leq M \leq 40$ ). Pe fiecare din cele  $N$  linii, sunt  $M$  cifre egale cu 0 dacă este aer sau cu 1 dacă este gheață pe acea poziție. Pe margini este sigur aer.

Fișierul 'aisberg.out' conține pe prima linie numărul de unități de timp în care se va topi toată gheața ( $T$ ).

Următoarele  $T$  linii vor conține pe fiecare linie  $i$  numărul de unități de gheață existente la începutul intervalului de timp  $i$ .

Exemplu: aisberg.in

```
6 7
0 0 0 0 0 0 0
0 1 1 1 1 1 0
0 0 1 1 1 0 0
0 0 1 1 1 0 0
0 1 1 1 1 1 0
0 0 0 0 0 0 0
```

aisberg.out

```
4
16
12
8
2
```

8. (Valoare maximă - \*\*\*\*) Fie șirurile  $a[1], a[2], a[3], \dots, a[n]$  și  $b[1], b[2], b[3], \dots, b[m]$ ,  $m > n$ . Să se maximizeze valoarea expresiei  $E = a[1]x[1] + a[2]x[2] + \dots + a[n]x[n]$ , unde  $x[i]$  sunt elemente ale șirului  $b$ .

Datele de intrare se vor citi din fișierul valmax.in în formatul următor: pe prima linie numerele  $n$  și  $m$ , iar pe următoarele două linii elementele celor două șiruri. Rezultatul va fi afișat pe ecran.

Exemplu: Pentru  $n=6$ ,  $m=8$  și șirurile  $a=(3, 7, -10, 5, -1, 2)$  respectiv  $b=(10, 5, 20, -20, -2, 7, 9, -10)$ , valoarea maximă a lui  $E$  este 441

9. (Globulețe - \*\*\*) O zonă dreptunghiulară este împărțită în  $n \times m$  parcele, câte  $m$  pe o linie. În fiecare parcelă există un brăduț, plin cu globulețe. Gigel își alege un brăduț, dar el vrea ca acesta să aibă cât mai multe globulețe. De aceea el va lua toate globulețele din brazi situati în parcelele învecinate la N,S,V,E de parcela în care el se află și le va pune în bradul ales inițial.

Exemplu: Dacă Gigel va alege brăduțul de la parcela de pe linia 3 și coloana 3, atunci bradul din acea parcelă va avea 40 de globulețe, iar cei patru brăduți învecinați vor rămâne fără nici unul.

2	1	6	17
3	4	8	6
1	3	3	8
14	4	3	2

2	1	6	17
3	4	0	6
1	0	40	0
14	4	0	2

Gigel mai are însă  $k-1$  frați. Toți vor dori ca brazi lor să fie împodobiți cu cât mai multe globuri.

Realizați un program care să identifice care brăduți să fie aleși de cei  $k$  copii astfel încât numărul total de globulețe din brăduții aleși de ei să fie maxim. Fișierul text glob.in va conține pe prima linie numerele  $n$ ,  $m$  și  $k$  despărțite prin câte un spațiu. Pe următoarele linii se găsesc câte  $m$  numere naturale mai mici decât 50000 reprezentând globulețele din brăduții situați pe fiecare linie în parte. ( $n, m, k \leq 100$ ). Rezultatele vor fi scrise în fișierul glob.out. Pe prima linie numărul total de globulețe din brăduții aleși de copii, iar pe următoarele linii, linia și coloana fiecărui brăduț ales.

Exemplu:

glob.in

```
4 4 3
2 1 6 17
3 4 8 6
1 8 8 8
14 4 8 2
```

glob.out

```
88
1 4
3 3
4 1
```

10. (Domino - \*\*\*\*) O piesă de domino are formă dreptunghiulară împărțită în două părți egale (stânga-dreapta), pe fiecare dintre acestea fiind înscrisă o cifră între 0 și 6. Două piese alăturate formează o secvență validă dacă cifrele care au devenit vecine sunt fie egale fie complementare (au suma egală cu 6);

Exemplu de secvențe valide de lungime 2 (formate prin alăturarea a două piese).

2	4	2	1
---	---	---	---

3	4	4	5
---	---	---	---

Gigel are pe masă un șir de  $n$  piese de domino. El va extrage o piesă din șir și o așează în aceeași poziție (fără să o rotească) la finalul șirului de piese. El speră astfel ca șirul nou creat va conține o secvență validă de piese de lungime maximă.

Identificați care piesă trebuie mutată la finalul șirului pentru a obține astfel un șir cu o subsecvență validă de lungime maximă.

Din fișierul text domino.in se citește de pe prima linie numărul  $n$  de piese din șir. Pe a doua linie sunt scrise  $2*n$  cifre ce se găsesc înscrise pe piesele de domino, în ordine de la stânga la dreapta. ( $n \leq 10000$ )

În fișierul text domino.out se va scrie pe prima linie două numere, reprezentând numărul de ordine al piesei din șir care este extrasă și mutată la finalul șirului, și lungimea maximă a secvenței valide determinate. Soluția nu este unică, dar se va afișa una singură.

Exemplu:

domino.in		domino.out
4	1 2	
2 3 4 5 4 5 3 4		

11. (Râma - \*\*\*\*) O râma se mișcă într-o zonă pătratică, intrând și ieșind de sub pământ. Ea înaintază spre centrul zonei, plecând de la suprafață din (1,1) și mergând paralel cu cele patru laturi (fără să treacă printr-un loc de două ori), descriind astfel o spirală. În figura următoare exemplificăm ordinea la deplasare în situația unei zone pătratice de 4 linii și 4 coloane.

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7

Deplasarea se face înaintând alternativ fie la suprafața pământului fie pe sub pământ. La întâlnirea unei gropi de pe traseu, râma va intra în groapă dacă deplasarea se făcea la acel moment la suprafață sau va ieși la suprafață prin acea groapă, dacă înaintarea se făcea pe sub pământ. Pe hartă gropile sunt codificate cu valoarea 0, restul valorilor fiind codificate cu 1. La coordonata (1,1) nu se poate afla o groapă. Realizați un program prin care sunt identificate gropile pe care râma le-a folosit pentru a ieși la suprafață. Acestea vor fi enumerate în ordinea întâlnirii lor pe taseu.

În fișierul rama.in se găsește pe prima linie numărul  $n$  reprezentând numărul de linii și coloane al zonei în care se plimbă râma.

Pe următoarele  $n$  linii este descrisă harta zonei prin  $n$  cifre binare {0,1}, despărțite fiecare prin câte un spațiu.

În fișierul rama.out se află pe prima linie numărul de gropi identificate, iar pe următoarele  $p$  linii câte o pereche de numere despărțite printr-un spațiu reprezentând coordonatele gropilor, în ordinea întâlnirii lor pe traseu.

Exemplu:

rama.in		rama.out		rama.in		rama.out
4	2		4		3	
1 1 0 1	3 4		1 1 1 0		2 4	
1 1 1 1	4 1		1 0 1 0		4 2	
1 1 1 0			1 1 0 1		2 2	
0 0 1 1			0 0 0 1			

12. (Jocul Domino -\*\*\*\*) În acest joc se folosesc piese dreptunghiulare de aceleași dimensiuni. Fața unei piese este împărțită printr-o linie în două pătrate marcate printr-un număr de puncte (0..6). Se consideră un șir de  $n$  piese de domino. El se consideră bine aranjat dacă pentru orice două piese așezate consecutiv, pătratele lor alăturate sunt marcate fie cu același număr de puncte fie suma acestora este egală cu 6. Exemplu de șir de domino bine aranjat (0,2),(2,5),(1,3)

Trebuie să realizați un program care determină dacă se poate obține un șir bine ordonat de piese de domino, având voie să rotiți piesele, dar nu să le schimbați locul în cadrul șirului.

Afișați fie șirul bine ordonat, fie mesajul 'Imposibil'. În fișierul de intrare domino.in se găsesc un număr par de cifre mai mici sau egale cu șase, două câte două valori reprezentând marcasele de pe o piesă.

Exemplu:

domino.in		domino.out
2 3 4 5 6 5 0 3	3 2 4 5 5 6 0 3	

13. (Etalon - \*\*\*\*) Se consideră o mulțime de  $N$  etaloane de greutate cunoscute folosite pentru cântărirea cu ajutorul unui taler. Scrieți un program care determină numărul total de greutate care pot fi cântărite folosind etaloanele date.

Fișierul etalon.in va conține pe prima linie numărul  $N$  ( $N \leq 200$ ), iar pe a doua linie cele  $N$  greutăți ale etaloanelor date (valori  $\leq 100000$ ). Rezultatul se va afișa pe prima linie în fișierul etalon.out.

Exemplu:

etalon.in		etalon.out
2	3	
5 1	Deoarece se pot măsura greutățile: 1,5 și 6.	

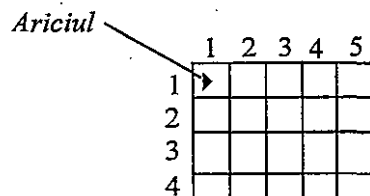
14. (Numere super-prime - \*\*\*\*) Se consideră secvența de numere prime  $P_1, P_2, \dots, P_n, \dots$ . Un număr este super prim dacă este prim și dacă numărul lui de ordine în șirul numerelor prime este un număr prim. De exemplu 3 este super prim (stă pe poziția a 2-a), dar 7 nu este (poziția a 4-a).

Realizați un program care descompune un număr dat ca sumă de numere super prime. Dacă există mai multe posibilități se va afișa cea cu număr minim de termeni. Numărul  $N$  nu va depăși 10000 și se va citi din fișierul super.in.

Fișierul super.out va conține pe o singură linie termenii sumei separați prin câte un spațiu.

Exemplu: Pentru  $N=6$  fișierul super.out va conține 3 3.

15. (Ariciul - \*\*\*\*) Planul unei livezi de formă dreptunghiulară cu dimensiunile  $n \times m$  este format din zone pătrate cu latura 1 (vezi desenul). În fiecare zonă crește un pom. Din fiecare pom în zona respectivă pot cădea jos câteva mere. În zona stânga-sus se află un arici. Ariciul dorește să ajungă în zona dreapta-jos. În livadă există restricții de deplasare: ariciul se poate mișca din zona curentă în zona vecină din dreapta sau de jos. Elaborați un program care determină numărul maxim de mere pe care le poate strânge ariciul deplasându-se în zona dorită.



Planul livezii este redat prin tabloul  $A$  cu  $n$  linii și  $m$  coloane ( $2 \leq n, m \leq 100$ ). Elementul  $A[i,j]$  al acestui tablou indică numărul de mere căzute din pom în zona cu coordonatele  $(i,j)$ . Fișierul text arici.in conține pe prima linie numerele  $n, m$  separate prin spațiu. Pe fiecare din următoarele  $n$  linii conțin câte  $m$  numere separate prin spațiu. Nici o valoare de pe linie nu depășește 1000. Fișierul text arici.out conține o singură linie pe care se scrie numărul maxim de mere, strânse de arici.

Exemplu:

arici.in	7	arici.out
3 3		
0 4 1		
0 1 1		
1 0 1		

16. (Paranteze - \*\*\*) Se considera șiruri de  $2 \cdot n$  ( $n \leq 501$ ) paranteze rotunde închise sau deschise. Un șir de paranteze se numește valid dacă în orice poziție numărul parantezelor deschise până la acea poziție este mai mare sau egal cu numărul parantezelor închise până la acea poziție. Să se determine câte șiruri de paranteze valide există.

Fișierul de intrare p.in conține numărul  $n$ .

Fișierul de ieșire p.out trebuie să conțină o singură linie pe care se va afla un singur număr care reprezintă câte șiruri de paranteze valide de lungime  $2 \cdot n$  există.

Exemplu:

p.in	5	p.out
3		

17. (Statistică - \*) Se consideră un șir de  $n \leq 10001$  numere naturale. Să se realizeze un grafic pe verticală în ordine descrescătoare a numărului de apariții a valorilor din șir.

Exemplu: Pentru  $n=12$  și valorile : 3, 4, 12, 5, 4, 2, 5, 3, 3, 3, 12, 5 se va afișa :

3	5	4	12	2
*				
*	*			
*	*	*	*	
*	*	*	*	*

18. (Subșir crescător - \*\*\*) Se consideră un șir de  $N$  numere naturale. Se cere să se determine cel mai lung subșir strict crescător al șirului, cu proprietatea că toate elementele sale sunt numere prime.

Pe prima linie a fișierului subsir.in se află  $N$  ( $1 \leq N \leq 2000$ ). Pe următoarea linie se află elementele șirului, valori întregi din intervalul  $[2, 30000]$ .

În fișierul subsir.out se va afișa pe prima linie lungimea subșirului cerut. Pe următoarea linie se vor scrie elementele subșirului despărțite prin câte un spațiu.

Exemplu:

subsir.in	5	subprim.out
10		
2 5 3 7 7 9 11 8 6 13		2 3 7 11 13

19. (Pătrat - \*\*\*\*) Fie o matrice  $A$ , de dimensiuni  $N \times M$ , ale cărei elemente pot fi 0 sau 1. Numim pătrat o mulțime de elemente  $A[i,j]$  ce formează un subtablou laturile egale. Să se determine aria maximă a unui pătrat din matricea  $A$ .

Valorile  $N$  și  $M$  ( $1 \leq N; M \leq 200$ ) se vor citi de pe prima linie a fișierului patrat.in. Pe fiecare din următoarele  $N$  linii se afla câte  $M$  valori din mulțimea  $\{0,1\}$ , neseparate prin spații, reprezentând elementele matricei  $A$ .

În fișierul patrat.out se va afișa valoarea ariei corespunzătoare pătratului maximal.

Exemplu:

patrat.in	9	patrat.out
8 8		
01011101		
10111101		
00111010		
11111111		
10111010		
11110111		
01011111		
11111111		

20. (Romb - \*\*\*\*) Fie o matrice  $A$ , de dimensiuni  $N \times M$ , ale cărei elemente pot fi 0 sau 1. Numim romb o mulțime de elemente  $A[i,j]$  ce formează un subtablou cu proprietățile  $A[i,j]=1$  și  $|i-X|+|j-Y| \leq R$ , unde  $(X, Y)$  reprezintă centrul rombului, iar  $R$  raza lui. Să se determine raza maximă a unui romb din matricea  $A$ .



Valorile  $N$  și  $M$  ( $1 \leq N; M \leq 200$ ) se vor citi de pe prima linie a fișierului romb.in. Pe fiecare din următoarele  $N$  linii se afla câte  $M$  valori din mulțimea  $\{0,1\}$ , neseperate prin spații, reprezentând elementele matricei  $A$ .

În fișierul romb.out se va afișa valoarea  $R$  corespunzătoare rombului maximal.

Exemplu:

romb.in		romb.out
8 8	2	
01011101		
10111101		
00111010		
11111111		
10111010		
11110111		
01011111		
11111111		

21. (Partiționare in trei - \*\*\*) Fie un șir cu  $N$  elemente naturale. Acesta trebuie partiționat în trei secvențe de elemente consecutive. Fiecare secvență va fi caracterizată de suma elementelor sale. Să se partiționeze șirul astfel încât diferența dintre suma maximă și suma minimă să fie cât mai mică.

De pe prima linie a fișierului trei.in se va citi  $N$  ( $10 \leq N \leq 32000$ ). Pe următoarea linie se vor citi elementele șirului, numere naturale din intervalul  $[1..30000]$ .

În fișierul trei.out se va afișa diferența minimă dintre suma maximă și suma minimă.

Exemplu:

trei.in		trei.out
10	3	
4 7 1 2 2 3 9 3 4 4		

22. (Suma - \*\*\*) Vom considera un șir cu  $N$  elemente și o valoare întreagă  $M$ . Să se determine câte perechi de elemente distincte există cu proprietatea că suma lor este  $M$ . Pe prima linie a fișierului suma.in se afla  $N$  și  $M$  ( $1 \leq N \leq 10.000$ ;  $1 \leq M \leq 1.000.000.000$ ). Pe următoarea linie se află elementele șirului, valori întregi din intervalul  $[1, 1.000.000.000]$ . În fișierul suma.out se va afișa numărul de perechi care respectă condiția din enunț.

Exemplu:

suma.in		suma.out
8 10	3	
7 2 3 5 4 6 1 8		

23. (Puncte - \*\*\*) De ziua lui, Gigel a primit un poligon convex cu  $N \leq 65536$  vârfuri. Fiindcă nu știa ce să facă cu el, s-a apucat să tragă linii între oricare două vârfuri neadiacente. După ce a tras toate liniile posibile a observat că oricare trei linii nu se intersectează în același punct. Fiind o persoană curioasă, el ar vrea să știe câte puncte de intersecție există în interiorul poligonului.

Din fișierul puncte.in se citește numărul  $N$ , iar în fișierul puncte.out se scrie rezultatul.

Exemplu:

puncte.in		puncte.out
5	5	
		(http://infoarena.devnet.ro)

24. (Zile de naștere - \*\*\*\*\*) Într-o cameră se află  $N \leq 51$  persoane. Fiecare persoană este născută într-una din cele  $Z \leq 366$  zile ale unui an.

Determinați zilele de naștere ale fiecărei persoane, astfel încât în cameră să existe  $K$  perechi de persoane născute în aceeași zi.

În fișierul days.in se află numerele întregi  $N, Z$  și  $K$ , separate prin câte un spațiu.

În fișierul days.out veți afișa o singură linie, care conține  $N$  valori întregi, cuprinse între 1 și  $Z$ , reprezentând zilele de naștere ale celor  $N$  persoane, astfel încât în cameră să existe  $K$  perechi de persoane născute în aceeași zi. Dacă există mai multe soluții, puteți afișa oricare dintre ele. Dacă nu există nici o soluție, atunci afișați în fișier numai valoarea 0.

Exemplu:

days.in		days.out
5 365 4	1 1 1 2 2	
		(http://infoarena.devnet.ro)

25. (Secvența - \*\*\*\*\*) Gigel are o secvență de  $N \leq 200.000$  numere întregi din intervalul

$[-10.000, 10.000]$  și vrea să găsească un subșir de sumă maximă cu proprietatea că oricare două elemente ale subșirului nu sunt aflate pe poziții consecutive în secvență.

În fișierul secv.in se va găsi numărul  $N$  și apoi  $N$  numere întregi, iar în fișierul secv.out suma subșirului cerut.

Exemplu:

secv.in		secv.out
7	16	
3 7 5 -1 6 6 2		(http://infoarena.devnet.ro)

26. (Tester - \*\*\*\*\*) Ion și Vlad s-au gândit să facă  $M$  ( $M < 100$ ) interschimbări într-un vector  $V$  de  $N < 19$  elemente. O interschimbare constă în alegerea a două poziții  $i$  și  $j$  ( $0 < i, j < N+1$ ) și schimbarea valorilor  $V[i]$  și  $V[j]$  între ele doar dacă  $V[i] > V[j]$ . Pe baza setului de interschimbări determinați dacă se sortează vectorul  $V$ , indiferent de valorile care le conține.

Pe prima linie din fișierul tester.in se găsesc numerele  $M$  și  $N$ . Pe următoarele  $M$  linii se găsesc perechi de numere  $i$  și  $j$ . Pe prima linie din fișierul tester.out se va scrie "DA" dacă cele  $M$  interschimbări sortează orice vector, și "NU" în caz contrar.

Exemplu:

<p>tester.in</p> <pre>6 4 1 2 2 3 3 4 1 2 2 3 1 2</pre>	<p>DA</p>	<p>tester.out</p>
---	-----------	-------------------

(<http://infoarena.devnet.ro>)

27. (Timbre - \*\*\*\*) Fiind date un set de  $n$  valori distincte de timbre și limita superioară  $k$  a numărului de timbre care pot fi lipite pe un plic, determinați cea mai mare secvență de valori consecutive de la 1 la  $M$  cenți care se poate obține.

Datele de intrare se citesc din fișierul timbre.in ce conține:

- pe prima linie din fișier se afla  $k$  ( $k \leq 200$ ), numărul total de timbre ce pot fi folosite și  $n$  numărul de valori ale timbrelor,  $n \leq 50$ ; aceste valori sunt mai mici decât 10000

- pe următoarea linie se găsesc cele  $n$  valori ale timbrelor separate prin câte un spațiu.

Datele de ieșire se vor scrie în fișierul timbre.out care va conține un singur număr reprezentând numărul  $M$  (maxim) de valori consecutive care se pot forma cu maxim  $k$  timbre de valori date.

Exemplu:

<p>timbre.in</p> <pre>5 2 1 3</pre>	<p>13</p>	<p>timbre.out</p>
-------------------------------------	-----------	-------------------

28. (Frații - \*\*\*\*\*) Gigel, într-o zi când își făcea temele la matematică, s-a apucat să scrie pe o foaie de hârtie, un șir de fracții ireductibile de forma  $\frac{p}{q}$  cu  $1 \leq p, q \leq N$ , unde  $N < 1.000.001$  este un număr natural ales de el. De exemplu, pentru  $N = 4$  el a obținut următorul șir:

$\frac{1}{1} \frac{1}{2} \frac{1}{3} \frac{1}{4} \frac{2}{1} \frac{2}{3} \frac{3}{1} \frac{3}{2} \frac{3}{4} \frac{4}{1} \frac{4}{3}$

Gigel s-a apucat apoi să numere câte fracții a obținut pentru  $N = 4$  și a văzut că sunt 11.

Fiind dat un număr natural  $N$ , să se determine câte fracții sunt în șirul de fracții construit după regulile de mai sus.

Fișierul de intrare fractii.in conține pe prima linie numărul natural  $N$ .

Fișierul de ieșire fractii.out trebuie să conțină un număr natural pe prima linie care reprezintă câte fracții sunt în șir.

Exemplu:

<p>fractii.in</p> <pre>3</pre>	<p>7</p>	<p>fractii.out</p>
--------------------------------	----------	--------------------

(<http://infoarena.devnet.ro>)

29. (Cifra - \*\*\*\*) Gigel, fiind plictisit, se juca în timpul orei de matematică, desenând pe o foaie. Din păcate, profesorul l-a văzut și i-a spus că îi pune nota 4 dacă nu rezolvă următoarea problemă: pentru o valoare  $N$  dată trebuie să determine ultima cifră a sumei  $1^1 + 2^2 + \dots + N^N$ .

Scrieți un program care să-l ajute pe Gigel și să determine ultima cifră a acestei sume pentru  $T$  valori date ale lui  $N$ .

Pe prima linie din fișierul cifra.in se va afla numărul  $T \leq 30.000$ . Pe următoarele  $T$  linii se vor găsi valori ale lui  $N < 10^{100}$  pentru care trebuie găsit răspunsul.

Pe cele  $T$  linii ale fișierului cifra.out se vor găsi răspunsurile pentru valorile lui  $N$  date în fișierul de intrare.

Exemplu:

<p>cifra.in</p> <pre>5 1 2 3 4 5</pre>	<p>1 5 2 8 3</p>	<p>cifra.out</p>
--	------------------	------------------

(<http://infoarena.devnet.ro>)

30. (Permutări - \*\*\*\*\*) O permutare de lungime  $N < 201$  este un șir de elemente distincte din mulțimea  $\{1, 2, 3, \dots, N\}$ . Spunem că o permutare are  $K$  maxime dacă există fix  $K$  poziții distincte în permutare, pentru care elementul curent este mai mare decât toate elementele din stânga lui. Scrieți un program care determină câte permutări de lungime  $N$  cu  $K$  maxime există.

Pe prima linie a fișierul perm.in se vor găsi numerele  $N$  și  $K$ , separate prin câte un spațiu, iar pe prima linie a fișierul perm.out se va găsi numărul de permutări de lungime  $N$  cu  $K$  maxime.

Exemplu:

<p>perm.in</p> <pre>5 3</pre>	<p>35</p>	<p>perm.out</p>
-------------------------------	-----------	-----------------

(<http://infoarena.devnet.ro>)

31. (Secvența - \*\*\*\*\*) Gigel are un șir de  $N \leq 500.000$  numere întregi din intervalul

$[-30.000, 30.000]$ . Toată lumea știe că o secvență este un subșir de numere care apar pe poziții consecutive în șirul inițial. Gigel a definit baza unei secvențe ca fiind minimul valorilor elementelor din secvența respectivă.

Fiind dat un număr natural  $K$ , determinați pentru Gigel o secvență de lungime cel puțin  $K$  cu baza maximă.

Fișierul de intrare secventa.in conține pe prima linie numerele  $N$  și  $K$ , separate prin spațiu. Pe cea de a doua linie se află elementele șirului separate prin câte un spațiu.

Fișierul de ieșire secventa.out trebuie să conțină o singură linie cu trei numere: poziția de început și de sfârșit a secvenței de lungime cel puțin  $K$  cu baza maximă și valoarea maximă a bazei.

Exemplu:

secventa.in  
8 3  
-1 2 3 1 0 4 8 6

secventa.out  
6 8 4

(<http://infoarena.devnet.ro>)

32. (Joc - \*\*\*\*\*) Gicu și Nicu, olimpici la informatică și buni prieteni, mereu încearcă să îmbine activitățile lor cu informatica. Spre exemplu, când se plictisesc în ore ei joacă un joc, bazat pe următoarele reguli:

- fie o matrice cu numere întregi cuprinse în intervalul  $[-1.000, 1.000]$ , cu  $N$  linii și  $M$  coloane ( $N, M \leq 1.000$ )
- liniile sunt numerotate de la 1 la  $N$ , iar coloanele de la 1 la  $M$
- fiecare jucător mută alternativ un jeton plasat pe un element din matrice
- o mutare constă în plasarea jetonului pe o altă poziție și adăugarea valorii din matrice de pe poziția respectivă la scorul jucătorului care a făcut mutarea; odată plasat jetonul pe o poziție, jucătorul următor poate să mute jetonul doar pe o altă poziție din dreptunghiul format de colțul stânga-sus și poziția curentă a matricei
- jocul se termină când un jucător ajunge cu jetonul în colțul stânga-sus al matricei
- la începutul jocului, ambii jucători au scor 0, iar jucătorul care începe alege poziția inițială a jetonului

Presupunând că fiecare din cei doi joacă optim (prin joc optim se înțelege că Gicu va încerca să maximizeze diferența de scor, în timp ce Nicu va încerca să o minimizeze), și că Gicu va începe jocul, determinați poziția inițială a jetonului, astfel încât diferența de scor între Gicu și Nicu să fie maximă!

Prima linie a fișierului joc.in conține două numere întregi  $N$  și  $M$ , separate prin câte un spațiu, care reprezintă numărul de linii și coloane ale matricii. Următoarele  $N$  linii conțin câte  $M$  numere întregi, separate prin câte un spațiu, care descriu matricea.

Fișierul joc.out va conține trei numere întregi separate prin câte un spațiu: diferența maximă de scor între Gicu și Nicu și linia și coloana unde se va plasa jetonul la începutul jocului.

Exemplu:

joc.in  
1 6  
2 1 3 4 0 5

joc.out  
3 1 6

(<http://infoarena.devnet.ro>)

33. (Loto - \*\*\*\*\*) Gigel este un mare pasionat al jocurilor de noroc, iar cel mai mult îi place să joace la loto „6 din  $N$ ”. La acest joc, el poate scrie pe un bilet 6 numere, din  $N \leq 100$  numere naturale distincte date de Loteria Națională; un număr poate fi folosit pe un bilet de mai multe ori.

Gigel a visat într-o noapte că suma numerelor scrise pe biletul câștigător va fi  $S \leq 600.000.000$ , așa că a doua zi s-a dus să pună și el un bilet câștigător! Scrieți un program care îi spune lui Gigel ce numere trebuie să aleagă ca să obțină un bilet câștigător (cu suma  $S$ ).

Pe prima linie din fișierul loto.in se vor găsi numerele naturale  $N$  și  $S$ , separate prin câte un spațiu. Pe a doua linie vor fi  $N$  numere naturale distincte, date de Loteria Națională.

În fișierul loto.out se vor găsi 6 valori reprezentând numerele alese pentru biletul lui Gigel. Dacă nu se poate obține un bilet câștigător în fișierul de ieșire se va afla doar numărul -1.

Exemplu:

loto.in  
3 13  
1 2 3

loto.out  
1 1 2 3 3 3

(<http://infoarena.devnet.ro>)

34. (Secvența 2 - \*\*\*) Gigel s-a decis să devină olimpic la informatică, poate așa va reuși să-și rezolve singur problemele, și nu va mai cere ajutorul vostru! La ora de informatică, profesoara lui i-a dat să rezolve problema secvenței de sumă maximă: „Gigele, eu îți dau un șir  $N \leq 50.000$  numere întregi din intervalul  $[-25.000, 25.000]$ , iar tu trebuie să găsești o secvență (adică un subșir de numere care apar pe poziții consecutive în șirul inițial) cu suma elementelor maximă!”. După vreo 30 de minute, Gigel s-a ridicat mândru și a zis: „Am găsit algoritmul de complexitate optimă, doamna profesoară!”. Ca temă pentru acasă Gigel are de rezolvat aproape aceeași problemă: trebuie să găsească secvența de sumă maximă de lungime cel puțin  $K$ .

Gigel încă nu știe destul de multă informatică ca să poată rezolva această problemă, dar poate îl ajutați voi!

Scrieți un program care rezolvă problema din tema lui Gigel.

Fișierul de intrare secv2.in conține pe prima linie numerele  $N$  și  $K$ , separate prin spațiu. Pe cea de a doua linie se află elementele șirului separate prin câte un spațiu. Fișierul de ieșire secv2.out trebuie să conțină o singură linie cu trei numere: poziția de început și de sfârșit a secvenței de sumă maximă de lungime cel puțin  $K$  și suma secvenței.

Exemplu:

secv2.in  
8 3  
0 -6 2 1 4 -1 3 -5

secv2.out  
3 7 9

(<http://infoarena.devnet.ro>)

35. (Zăhărel - \*\*\*\*\*) Zăhărel este un mare pasionat al culorilor, astfel încât a luat o foaie de mate cu  $N$  linii și  $N$  coloane ( $6 \leq N \leq 1000$ ) și a desenat  $M$  ( $2 \cdot N \leq M \leq 10000$ ) buline roșii sau albastre, în căsuțele foi de mate, în diferite poziții.

După ce a desenat punctele a observat că există cel puțin un punct roșu pe fiecare linie și cel puțin un punct albastru pe fiecare coloană și astfel și-a pus următoarea problemă:

poate să construiască două poligoane (nu neapărat convexe) care să aibă același număr de vârfuri, unul din poligoane să aibă în vârfuri doar buline roșii, iar celălalt doar buline albastre, iar centrul de greutate al celor două poligoane să fie același?

Zăhărel nu este băiat pretențios deci n-are nimic împotriva dacă cele două poligoane se intersectează sau dacă sunt unul în interiorul celuilalt! Trebuie să fie respectate doar condițiile menționate mai sus... Reamintim că Zăhărel consideră centrul de greutate al unui poligon cu vârfurile  $(x_1, y_1) \dots (x_n, y_n)$  ca fiind punctul  $((x_1 + \dots + x_n)/n, (y_1 + \dots + y_n)/n)$ . Scrieți un program care, pentru o foaie de matematică desenată ca mai sus de Zăhărel, determină cele două poligoane.

Pe prima linie din fișierul zaharel.in se găsesc numerele naturale  $N$  și  $M$ . Următoarele  $M$  linii sunt de forma  $i j c$  unde  $i$  și  $j$  sunt numere naturale reprezentând linia, respectiv coloana unei buline, iar  $c$  este un caracter reprezentând culoarea (R pentru roșu și A pentru albastru)

Pe prima linie se va afișa un număr, reprezentând câte vârfuri are fiecare poligon. Pe următoarea linie se vor afișa punctele care descriu poligonul cu vârfurile în buline roșii, într-o ordine oarecare. Pe a treia linie se vor afișa punctele care descriu poligonul cu vârfurile în buline albastre, într-o ordine oarecare. Dacă nu există soluție, se va afișa -1 în fișierul de ieșire.

Exemplu: zaharel.in

```
6 12
1 3 R
2 4 R
3 1 R
4 6 R
5 2 R
6 4 R
2 1 A
4 2 A
3 3 A
1 4 A
6 5 A
6 6 A
```

zaharel.out

```
3
1 3 2 4 3 1
1 4 2 1 3 3
```

(<http://infoarena.devnet.ro>)

36. (Trapez - \*\*\*\*) Zăhărel este un tip care se plictisește repede la școală. Într-o zi cu soare, când n-avea chef să asculte ce predă profesorul de matematică s-a apucat să deseneze puncte pe o foaie de matematică. El a desenat  $N \leq 1.000$  astfel de puncte și apoi și-a pus următoarea întrebare: câte trapeze se pot forma cu vârfurile în aceste puncte? (doar era la ora de matel!). Un trapez este un patrulater convex cu cel puțin două laturi paralele.

Ajutați-l pe Zăhărel să determine câte trapeze poate forma cu cele  $N$  puncte de pe foaia de matematică, știind că oricare trei puncte sunt necoliniare.

Pe prima linie din fișierul de intrare trapez.in se găsește numărul natural  $N$ . Pe următoarele  $N$  linii se găsesc perechi de numere naturale reprezentând coordonatele punctelor, numere întregi din intervalul  $[0, 2.000.000.000]$ .

Pe prima linie din fișierul de ieșire trapez.out se va găsi numărul de trapeze care se pot forma.

Exemplu:

trapez.in

```
5
0 0
0 1
1 4
2 0
3 1
```

trapez.out

1

(<http://infoarena.devnet.ro>)

37. (Subșir - \*\*\*\*\*) Zăhărel încearcă să o învețe pe prietena lui Eugenia informatică. Astăzi a învățat-o programare dinamică și anume a început cu problema celui mai lung subșir comun: dându-se două șiruri de lungime maxim 500, formate doar din litere mici, să se determine cel mai lung subșir comun al celor două șiruri. Un subșir al unui șir este format din caractere (nu neapărat consecutive) ale șirului respectiv, în ordinea în care acestea apar în șir.

Eugenia a înțeles rezolvarea problemei, dar i-a pus următoarea întrebare lui Zăhărel: câte subșiruri comune de lungime maximă distincte există pentru cele două șiruri? Două subșiruri sunt distincte dacă există cel puțin un caracter în unul din ele care diferă de caracterul din celălalt subșir de pe aceeași poziție.

Ajutați-l pe Zăhărel să determinați restul împărțirii numărului de subșiruri comune de lungime maximă distincte pentru două șiruri date, la numărul 666013.

Pe prima linie a fișierului de intrare subsir.in se găsește primul șir, iar pe a doua linie cel de-al doilea șir. Pe prima linie a fișierului de ieșire subsir.out se va găsi numărul cerut.

Exemplu:

subsir.in

```
banana
oana
```

subsir.out

1

(<http://infoarena.devnet.ro>)

38. (Numere prime - \*\*) Gheorghe a învățat la școală despre numere prime. A învățat că un număr este prim, dacă se divide doar cu 1 și cu el însuși (1 nu este considerat număr prim). A aflat că există algoritmi foarte eficienți care pot determina dacă un număr este prim sau nu, în timp chiar sub polinomial.

Din păcate acești algoritmi sunt foarte complicați și Gheorghe s-a gândit la o aproximare. Ideea lui este să considere un număr prim dacă nu se divide la primele  $K$  numere prime.

Demonstrează că ideea lui Gheorghe este doar o aproximare. Dându-se un număr  $K \leq 100.000$ , afla cel mai mic număr  $N$  care nu este divizibil cu primele  $K$  numere prime, dar nu este prim.

Pe prima linie din fișierul `prim.in` se va afla numărul  $K$ .

Pe prima linie a fișierului `prim.out` se va găsi numărul  $N$  căutat.

Exemplu:

<code>prim.in</code>	49	<code>prim.out</code>
3		

(<http://infoarena.devnet.ro>)

39. (Baze - \*\*\*) Există numere care au proprietatea că se scriu, în două baze diferite, prin trei cifre identice. De exemplu, numărul  $273_{(10)}$  în baza 9 se scrie  $333_{(9)}$  și în baza 16 se scrie  $111_{(16)}$ .

Concepeți un program care să determine toate numerele mai mici ca  $N < 32001$  care au această proprietate.

Fișierul de intrare `baze.in` va conține pe prima linie numărul  $N$ .

În fișierul de ieșire `baze.out` se vor scrie numerele determinate, fiecare pe câte un rând. Pentru fiecare număr se vor scrie, separate prin câte un spațiu, numărul în baza 10 și cele 2 baze în care numărul respectiv are proprietatea din enunț.

Exemplu:

<code>baze.in</code>	273 9 16	<code>baze.out</code>
300		

(Concurs "Grigore Moisil", Lugoj 2001, cls. VII-VIII)

40. (Secvență palindromică - \*\*\*) Să considerăm un șir de caractere, care pot fi doar litere mici ale alfabetului englez. Numim secvență palindromică o succesiune de litere din șir care are proprietatea palindromică (fie că o parcurgem de la stânga la dreapta, fie că o parcurgem de la dreapta la stânga, secvența este aceeași).

De exemplu, succesiunea de litere *cojoc* are proprietatea palindromică.

Scrieți un program care să determine cea mai lungă secvență palindromică dintr-un șir dat.

Fișierul de intrare se numește `sp.in` conține două linii:

$N$  – numărul de litere din șirul de intrare ( $< 20001$ )

$s_1 s_2 \dots s_N$  – șirul de  $N$  litere mici

Fișierul de ieșire `sp.out` conține:

*poz* – poziția de început a celei mai lungi secvențe palindromice

*lg* – lungimea celei mai lungi secvențe palindromice

Exemplu:

<code>sp.in</code>	9 5	<code>sp.out</code>
22		
anaareuncojocasafumos		

(Concurs "Grigore Moisil", Lugoj 2001, cls. VII-VIII)

41. (Diferențe - \*\*\*\*) Se dă un șir de  $N < 15001$  numere întregi (între  $-30000$  și  $30000$ ). O secvență a acestui șir este alcătuită din  $M$  elemente (numere) consecutive ( $0 < M \leq N$ ). Să se scrie secvența a cărei sumă a elementelor este minimă, în modul.

Din fișierul `dif.in` se citește de pe prima linie  $N$ , iar pe următoarele  $N$  linii se află elementele șirului mare (în ordine).

În fișierul `dif.out` se va scrie un singur număr reprezentând suma (în modul) minimă.

Exemplu: <code>dif.in</code>	1	<code>dif.out</code>
3		
2		
-3		
4		

42. (Pitici - \*\*\*) Se consideră  $n$  pitici, care stau aliniați în rând, fiecare cu fața spre spatele celui alt. Piticii au pe cap căciulițe roșii și negre. Piticii cu căciulițe roșii spun întotdeauna adevărul, în timp ce piticii cu căciulițe negre mint întotdeauna. Fiecare pitic este întrebat câte căciulițe roșii vede în fața sa. În funcție de răspunsul piticilor, trebuie să stabiliți ce culoare are căciulița lor.

Din fișierul `pitici.in` se citesc răspunsul piticilor. Formatul fișierului este: pe prima linie se află numărul de pitici  $n$  ( $n < 20000$ ). Pe următoarele linii se află perechi de câte două numere care reprezintă: număr pitic – răspunsul la întrebare (separate prin spațiu).

Răspunsul se va scrie în fișierul `pitici.out` pe o singură linie, fiind format dintr-o secvență de  $n$  caractere R sau N, R reprezentând culoarea roșie și N culoarea negru.

Exemplu:

<code>pitici.in</code>	RNNRN	<code>pitici.out</code>
5		
3 2		
4 1		
2 2		
5 3		
1 0		

43. (Bile - \*\*) Gigel are  $N$  cutii cu bile roșii, verzi și albastre. Într-o zi, el se hotărăște să strângă toate bilele în trei cutii: cele roșii într-una, cele verzi în alta și cele albastre în alta, diferită de primele două. În acest scop, el dorește să mute cât mai puține bile din cutiile în care sunt în cele în care vor ajunge. Ajutați-l!

Fișierul de intrare `bile.in` are următoarea structură:

$N$  – nr. de cutii;  $3 \leq N \leq 1000$ ;

$R1 V1 A1$  – câte bile roșii, verzi și albastre are în prima cutie

$R2 V2 A2$  – ... și tot așa pentru celelalte cutii

...

$RN VN AN$

Fișierul de ieșire `bile.out` are următoarea structură:

$NR$  – numărul de bile mutate

$CR CV CA$  – numerele cutiilor în care ajung bilele roșii, verzi, respectiv albastre.

Exemplu:

bile.in	bile.out
3	4
3 0 1	1 3 2
1 2 2	
0 5 0	

44. (**Dominante** - \*\*\*) Se consideră  $N$  puncte în plan ( $N \leq 10000$ ), cu coordonate întregi (între  $-30000$  și  $30000$ ). Dintre acestea unele sunt dominante, iar altele nu. Un punct se consideră dominant dacă la dreapta lui (cu coordonata  $X$  cel puțin egală cu a lui) nu există nici un punct mai înalt decât el (cu coordonata  $Y$  mai mare, strict mai mare).

Cerința voastră este să stabiliți câte din cele  $N$  puncte sunt dominante.

Fișierul `dom.in` conține pe prima linie numărul de puncte. Pe următoarele  $N$  linii se află coordonatele  $X$  și respectiv  $Y$  ale punctelor, separate printr-un spațiu.

În fișierul `dom.out` trebuie scrise câte din acestea sunt dominante.

Exemplu:

dom.in	dom.out
3	2
2 0	
0 1	
0 2	

45. (**Semn** - \*\*\*) Se dă un șir de  $N$  ( $0 < N < 101$ ) numere pozitive (mai mici de 100). Se cere să se înmulțească o parte din aceste numere cu  $-1$ , așa încât adunate (toate numerele) să dea un număr cât mai apropiat de 0, pozitiv (poate să fie și 0, aceasta fiind cea mai bună variantă, dacă e posibil).

Din fișierul `semn.in` citiți de pe prima linie  $N$ , iar de pe următoarele  $N$  linii cele  $N$  numere.

În fișierul `semn.out` scrieți un singur număr reprezentând suma cea mai mică ce se poate obține.

Exemplu:

semn.in	semn.out
3	0
2	
3	
5	

46. (**Mouse** - \*\*\*) Un experiment urmărește comportarea unui șoricel pus într-o cutie dreptunghiulară, împărțită în  $m \times n$  cămăruțe egale de formă pătrată. Fiecare cămăruță conține o anumită cantitate de hrană. Șoricelul trebuie să pornească din colțul (1,1) al cutiei și să ajungă în colțul opus, mâncând cât mai multă hrană. El poate trece dintr-o cameră în una alăturată (două camere sunt alăturate dacă au un perete comun), mâncând toată hrana din cămăruță atunci când intră și nu intră niciodată într-o cameră fără hrană. Stabiliți care este cantitatea maximă de hrană pe care o poate mânca și traseul pe care îl poate urma pentru a culege această cantitate maximă.

Fișierul de intrare `mouse.in` conține pe prima linie două numere  $m$  și  $n$  reprezentând numărul de linii respectiv numărul de coloane ale cutiei, iar pe următoarele  $m$  linii cele  $m \times n$  numere reprezentând cantitatea de hrană existentă în fiecare cămăruță, câte  $n$  numere pe fiecare linie, separate prin spații. Toate valorile din fișier sunt numere naturale între 1 și 100.

În fișierul de ieșire `mouse.out` se vor scrie pe prima linie două numere separate printr-un spațiu: numărul de cămăruțe vizitate și cantitatea de hrană maximă culeasă. Pe următoarele linii se va scrie un traseu posibil pentru cantitatea dată, sub formă de perechi de numere (linie coloană) începând cu 1 1 și terminând cu  $m n$ .

Exemplu:

mouse.in	mouse.out
2 4	7 21
1 2 6 3	1 1
3 4 1 2	2 1
	2 2
	1 2
	1 3
	1 4
	2 4

(Olimpiada Județeană de Informatică, 2002, cls. IX)

47. La Loteria Națională există  $N$  ( $N < 1000$ ) bile inscripționate cu numere naturale, nenule, distincte de cel mult 4 cifre. Șeful de la loterie primește o cutie în care se află cele 6 bile extrase la ultimă rundă, restul bilelor neextrase fiind puse într-un seif. Deoarece are o fire poznașă, el scoate din cutie bila pe care este înscris numărul cel mai mic și o păstrează în buzunarul hainei sale. În locul ei va pune o bilă neextrasă, aflată în seif, având numărul cel mai apropiat de aceasta. Apoi continuă operația și scoate din cutie și bila pe care este înscris numărul maxim extras inițial, pe care o va pune în celălalt buzunar al său. De asemenea o va înlocui cu o altă bilă neextrasă inițial, aflată în seif, având numărul cel mai apropiat de aceasta.

Realizați un program care afișează în ordine crescătoare numerele de pe bilele aflate în cutie după modificările făcute de șef.

Fișierul de intrare `loto.in` conține pe prima linie numărul natural  $N$ , pe a doua linie cele  $N$  numere naturale scrise pe bile, iar pe a treia linie cele 6 numere naturale scrise pe bilele extrase de angajații loteriei. Valorile scrise pe aceeași linie sunt separate prin spații.

În fișierul de ieșire `loto.out` se vor afișa pe prima linie, separate prin câte un spațiu, cele 6 numere obținute în cutie după modificare făcute de șef, în ordine crescătoare.

Exemplu:

loto.in	loto.out
12	1 3 4 6 9 26
3 4 6 7 8 9 2 1 10 18 22 26	
2 9 3 4 22 6	

(Olimpiada Județeană de Informatică, 2010, cls. IV)

## Indicații și răspunsuri

### Secțiunea 1.1.1

1 b),d),e)	16 c)	30 c)	44 a),b),d)	58 b)
2 b),d),f)	17 b)	31 a),d)	45 c)	59 b),c),d)
3 a),b),f)	18 b),c),f)	32 b),c),d)	46 b),c)	60 c)
4 a),f)	19 a),b),e),f)	33 a),b),e)	47 d),e),f)	61 b)
5 a),d),f)	20 b),c),e)	34 b)	48 b),c),d)	62 c)
6 a),d),f)	21 d)	35 a),b),d)	49 a)	63 b)
7 a),b),d),f)	22 b),c),f)	36 b)	50 a),c),d)	64 d)
8 b),e)	23 b),c),d),e)	37 a),d)	51 b),f)	65 a)
9 c),d),e)	24 a),c),f)	38 a),b)	52 c)	66 c)
10 a),c),d),e)	25 a),c)	39 a),d)	53 c),d)	67 c),d)
11 c)	26 d)	40 c),d)	54 c)	68 a),c)
12 b)	27 b)	41 c)	55 d)	69 a),b)
14 a),c),d),f)	28 b)	42 b),d)	56 c)	70 b),c),d)
15 d)	29 d)	43 a),c)	57 a)	71 b),c)

### Secțiunea 1.1.2

#### Soluție test 1

a) 36 68 respectiv -48 22;      b) 2 2;      c) 2 6;

d)

```

1 var a,b,c,d,x: integer;
2 begin
3   readln(a,b);
4   c:=a+b; d:=a*b;
5   if c>d then begin x:=c; c:=d;
6     d:=x; end;
7   if a mod 2=0 then write(c,' ',d)
8   else write(d,' ',c);
9   end.

```

```

#include <iostream.h>
int a,b,c,d,x;
void main() {
  cin>>a>>b;
  c=a+b;d=a*b;
  if (c>d) { x=c; c=d; d=x; }
  if (a%2==0) cout<<c<<' '<<d;
               else cout<<d<<' '<<c;
}

```

#### Soluție test 2

a) 589 2;      b) 0 -3;      c) 1 1;      d) 4.2 1.3;

e)

```

1 var a,b,w:integer; x,y:real;
2 begin
3   readln(x,y);
4   a:=trunc(x*y); b:=trunc(x/y);
5   if a<b then begin
6     w:=a; a:=b; b:=w;
7   end;
8   if x<>trunc(x) then
9     write(a,' ',b)
10  else write(b,' ',a);
11  end.

```

```

#include <iostream.h>
#include <math.h>
int a,b,w; double x,y;
void main() {
  cin>>x>>y;
  a=(int)(x*y); b=(int)(x/y);
  if (a<b) { w=a; a=b; b=w; }
  if (x!=floor(x))
    cout<<a<<' '<<b;
  else cout<<b<<' '<<a;
}

```

#### Soluție test 3

a) 348 190 respectiv 3 211;      b) 1000 200;  
c) Cifra zecilor aparține mulțimii {0..5} și cifra unităților {(b mod 10+1)..9};  
d)

```

1 var a,b:integer;
2 begin
3   readln(a,b);
4   if a mod 10 < b mod 10 then
5     a:=a - a mod 10 + b mod 10;
6   if a div 10 mod 10 > 5 then
7     a:=a - a div 10 * 10
8   else b:=b - a mod 100;
9   writeln(a,' ',b)
10  end.

```

```

#include <iostream.h>
int a,b;
void main() {
  cin>>a>>b;
  if (a%10<b%10) a-=a%10-b%10;
  if ((a/10)%10>5) a-=(a/10)*10;
               else b-=a%100;
  cout<<a<<' '<<b<<endl;
}

```

#### Soluție test 4

a) 41 31 7;  
b) Orice set de valori pentru care (s1+s2<60) și (m1+m2<60);  
c) Determinați măsura unghiului sumă dintre două unghiuri exprimate în grade, minute, secunde. Datele se citesc de la intrarea standard;

d)

```

1 var
2   s1,s2,m1,m2,g1,g2,s,m,g:integer;
3 begin
4   read(s1,s2,m1,m2,g1,g2);
5   s:=s1+s2; m:=m1+m2;
6   g:=g1+g2;
7   if s>60 then begin
8     s:=s mod 60; m:=m+1; end;
9   if m>60 then begin
10    m:=m mod 60; g:=g+1; end;
11  write(g,' ',m,' ',s) end.

```

```

#include <iostream.h>
int s1,s2,m1,m2,g1,g2,s,m,g;
void main() {
  cin>>s1>>s2>>m1>>m2>>g1>>g2;
  s=s1+s2; m=m1+m2; g=g1+g2;
  if (s>60) { s%=60; m++; }
  if (m>60) { m%=60; g++; }
  cout<<g<<' '<<m<<' '<<s;
}

```

#### Soluție test 5

a) "Exista numere negative" respectiv "Numere pozitive";  
b) Orice triplet de numere negative;

c)

```

1 daca (a<0) or (b<0) or (c<0) atunci
2   scrie 'Exista nr negativ'
3 altfel
4   scrie 'Numere Pozitive'
5

```

d)

```

1 var a,b,c:integer;
2 begin
3   read(a,b,c);
4   if a*b<0 then
5     write('Exista nr negativ')
6   else
7     if b*c<0 then
8       write('Exista nr negativ')

```

```

#include <iostream.h>
void main() {
  int a,b,c;
  cin>>a>>b>>c;
  if (a*b<0)
    cout<<"Exista nr negativ";
  else
    if (b*c<0)

```

Solutie test 6

```

1  daca ((a+b)/2=c) or ((a+c)/2=b) or ((c+b)/2=a) atunci
2      scrie 'Corect'
3  altfel
4      scrie 'Incorect'
5

```

e) Verificați dacă un triplet de numere citit, de la intrarea standard, reprezintă termenii unei progresii aritmetice

### Soluție test 7

```

1  [daca (a>b) atunci a <-> b
2  [
3  [daca (b>c) atunci c <-> b
4  [
5  [daca (a>c) atunci a <-> c
6  [

```

174

### Soluție test 8

```
c)
1  daca ((a+b)<c) or ((a+c)<b) or ((c+b)<a) atunci
2      scrie 'Nu'
3  altfel
4      scrie 'Corect'
5
```

### Solutie test 9

```
c)
1  var a,b,c:longint;
2  begin
3    read(a);
4    if a mod 100<50 then
5      a:=a - a mod 100
6    else a:=a + 100 - a mod 100;
7    writeln(a)
8  end.
```

### Soluție test 10

```

d)
1  var x:real;
2  begin
3      readln(x);
4      x:=x*10;
5      if trunc(x)mod 10<>0 then
6          x:=trunc(x)/10
7      else begin
8          x:=x*10;
9          if trunc(x)mod 10<>0 then
10             x:=trunc(x)/100

```



```

11 else
12     x:=x/100
13 end;
14 write(x:0:3)
15 end.
16
17 if ((int) x)%10)
18     x=floor(x)/100.0;
19 else x=x/100.0;
20 }
21 printf("%.3f\n",x);
22 }

```

### Sectiunea 1.1.4

```

17.
1 citeste p;
2 d ← p + 2*p + 4*p + 8*p ;
3 scrie d; stop.

18.
1 citeste n;
2 n ← n div 100 ;
3 scrie n, n div 10 + n mod 10; stop.

```

```

20.
1 citeste n;
2 scrie n div 100
3 scrie n div 10 mod 10
4 scrie n mod 10; stop.

```

```

22.
1 citeste x,y,z;
2 daca x=y atunci a ← 2*z
3 altfel
4     daca x*y<x+z atunci a ← x*y
5     altfel a ← x + z
6
7

```

```

23.
1 citeste n;
2 daca n mod 10< n div 10 mod 10 atunci
3     scrie n mod 10, n div 10 mod 10
4 altfel
5     scrie n div 10 mod 10, n mod 10
6

```

```

25.
1 citeste s1, s2, m1, m2, g1, g2;
2 s ← s1 + s2; m ← m1 + m2; g ← g1 + g2;
3 daca s>60 atunci
4     s ← s mod 60; m ← m + 1;
5
6     daca m>60 atunci
7         m ← m mod 60; g ← g + 1;
8
9 scrie g, m, s; stop.

```

```

27.
1 citeste n;
2 daca n mod 10 = n div 10 mod 10 atunci
3     scrie n + 1, -n+2
4 altfel
5     daca n mod 10 > n div 10 mod 10 atunci scrie n mod 10
6     altfel scrie n div 10 mod 10
7
8

```

```

28.
1 citeste n;
2 daca n mod 2 = 0 atunci
3     scrie n - 2, n + 2
4 altfel
5     scrie n - 1, n + 1
6

```

```

29.
1 citeste n, m;
2 daca n < m atunci
3     scrie n / m
4 altfel
5     scrie m / n
6

```

```

30.
1 citeste a, b, c, d, e, s;
2 s ← a div |a| + b div |b| + c div |c| + d div |d| + e div |e|
3 daca s > 0 atunci
4     scrie majoritatea pozitive
5 altfel
6     scrie majoritatea negative
7

```

### Sectiunea 1.2.1

1 b),d)	9 c)	17 b),d)	25 e)	33 b)
2 b),c)	10 b),d)	18 a),c)	26 b),c),d)	34 a),c)
3 a)	11 a),b)	19 b)	27 a),c)	35 a)
4 c),d)	12 b),c)	20 c)	28 b),c)	36 d)
5 b),c)	13 c)	21 d)	29 a)	37 d)
6 c),d)	14 c)	22 c),d)	30 b)	38 b)
7 c),d)	15 a),d)	23 a),b)	31 c)	39 a),c)
8 b)	16 d)	24 b),c),d)	32 d)	

## Secțiunea 1.2.2

### Soluție test 1

- a) 4 respectiv 5;  
b) Oricare două valori impare egale;  
c) Determinați numărul de numere pare aflate în intervalul  $[a, b]$ ;

```
d)
1  var a,b,c,i :integer ;
2  begin
3  readln(a,b);
4  c:=0;
5  for i:=a to b do
6    if i mod 2=0 then inc(c);
7  if c>0 then write(c)
8  else write('Nu exista');
9  end.
10
#include <iostream.h>
int a,b,c,i;
void main() {
  cin>>a>>b;
  c=0;
  for (i=a;i<=b;i++)
    if (i%2==0) c++;
  if (c>0) cout<<c;
  else cout<<"Nu exista";
}
```

### Soluție test 2

- a) 13;  
b) Orice șir de n numere pentru care toate valorile aparțin intervalului  $[n, 2*n]$ ;  
c) Orice șir de n numere pentru care toate valorile nu aparțin intervalului  $[n, 2*n]$ ;

```
d)
1  var n,x,nr,i :integer ;
2  begin
3  readln(n);
4  nr:=0;
5  for i:=1 to n do begin
6    read(x);
7    if (x<n) or (x>2*n) then
8      nr:=nr+x;
9  end;
10 write(nr);
11 end.
#include <iostream.h>
int n,x,nr,i;
void main() {
  cin>>n;
  nr=0;
  for (i=0;i<n;i++) {
    cin>>x;
    if (x<n || x>2*n) nr+=x;
  }
  cout<<nr;
}
```

### Soluție test 3

- a) 2, 4, 6;  
b) Orice șir de trei valori multipli de 3;  
c) Operația se efectuează de 6 ori;

```
d)
1  var x,s,i,j :integer ;
2  begin
3  for i:=1 to 3 do begin
4    read(x); s:=0;
5    for j:=1 to 2 do s:=s+ x;
6    write(s);
7  end;
8  end.
#include <iostream.h>
int x,s,i,j;
void main() {
  for (i=0;i<3;i++) {
    cin>>x; s=0;
    for (j=0;j<2;j++) s+=x;
    cout<<s;
  }
}
```

### Soluție test 4

- a) 6, 10 12, 8 5;  
b) 1, 1, 1, 1, 1;  
c) 15;

```
d)
1  var x, s, i, j :integer ;
2  begin
3  for i:=1 to 5 do begin
4    read(x); s:=0;
5    for j:=1 to i do s:=s+ x;
6    write(s);
7  end;
8  end.
#include <iostream.h>
int x,s,i,j;
void main() {
  for (i=1;i<=5;i++) {
    cin>>x; s=0;
    for (j=1;j<=i;j++) s+=x;
    cout<<s;
  }
}
```

### Soluție test 5

- a) 1, 2, 3, 4, 5, 6, 7, 8, 9;  
b) O singură dată;  
c) De n ori;

```
d)
1  var n,i, j :integer ;
2  begin
3  read(n);
4  for i:=1 to n do
5    for j:=1 to n do
6      write((i-1) *n+j);
7  end.
#include <iostream.h>
int n,i,j;
void main() {
  cin>>n;
  for (i=0;i<n;i++)
    for (j=0;j<n;j++)
      cout<<i*n+j+1;
}
```

### Soluție test 6

- a) 3; b) Orice șir de valori impare care se termină cu valoarea 0;  
c) Se citesc numere întregi până la întâlnirea valorii 0. Câte numere pare au fost introduse?

```
d)
1  var a, nr :integer ;
2  begin
3  read(a); nr:=0;
4  while a<>0 do begin
5    if a mod 2=0 then inc(nr);
6    read(a);
7  end;
8  write(nr);
9  end.
#include <iostream.h>
int a,nr;
void main() {
  cin>>a; nr=0;
  while (a!=0) {
    if (a%2==0) nr++;
    cin>>a; }
  cout<<nr;
}
```

### Soluție test 7

- a) 3 pentru ambele valori; b) Orice putere a lui 2.  
c) Câte cifre de 1 apar în scrierea binară a lui x

```
d)
1  var x, nr :integer ;
2  begin
3  read(x); nr:=0;
4  while x<>0 do begin
#include <iostream.h>
int x,nr;
void main() {
  cin>>x; nr=0;
```

```

5   if x mod 2=1 then inc(nr);
6   x:=x div 2;
7   end;
8   write(nr);
9   end.

```

```

while (x!=0) {
  if (x%2==1) nr++;
  x/=2;
}
cout<<nr;

```

#### Soluție test 8

- a) 3;  
 b) Orice șir de valori de parități diferite, introduse consecutiv, ce se termină cu o valoare impară;  
 c) Orice șir de valori de aceeași paritate, introduse consecutiv;  
 d)

```

1   var x, nr, y : integer ;
2   begin
3   read(x); nr:=0;
4   while x>0 do begin
5     read(y);
6     if x mod 2=y mod 2 then
7       inc(nr);
8     x:=y;
9   end;
10  write(nr); end.

```

```

#include <iostream.h>
int nr,x,y;
void main() {
  cin>>x; nr=0;
  while (x!=0) {
    cin>>y;
    if (x%2==y%2) nr++;
    x=y;
  }
  cout<<nr;
}

```

#### Soluție test 9

- a) 10; b) 1, 2, 3, 4;  
 c) Orice număr pentru care cifra unităților este 5, 6, 7, 8, 9;  
 d)

```

1   var x, nr : integer ;
2   begin
3   read(x); nr:=0;
4   repeat
5     inc(x);
6     if x mod 5=0 then inc(nr);
7   until nr=2;
8   write(x);
9   end.
10

```

```

#include <iostream.h>
void main() {
  int x,nr;
  cin>>x; nr=0;
  do {
    x++;
    if (x%5==0) nr++;
  } while(nr!=2);
  cout<<x;
}

```

#### Soluție test 10

- a) 2; b) Orice șir de x valori întregi ce nu aparțin intervalului [2,9]; c) 3;  
 d)

```

1   var x,nr,y:integer;
2   begin
3   read(x);nr:=0;
4   repeat
5     read(y);
6     if (y>1)and(y<10) then
7       inc(nr);
8     dec(x);
9   until x=0;
10  write(nr);
11  end.

```

```

#include <iostream.h>
int nr,x,y;
void main() {
  cin>>x; nr=0;
  do {
    cin>>y;
    if (y>1 && y<10) nr++;
    x--;
  } while (x!=0);
  cout<<nr;
}

```

#### Soluție test 11:

- a) 2, 6, 24, 120, 4; b) 1, 2, 0;

```

1   var p,i,x:integer;
2   begin
3   i:=0; p:=1; read(x);
4   while x>0 do begin
5     inc(i);p:=p*x;
6     read(x);write(p,' ');
7   end;
8   write(i);
9   end.
10

```

```

#include <iostream.h>
int i,p,x;
void main() {
  i=0; p=1; cin>>x;
  while (x) {
    i++; p*=x;
    cin>>x; cout<<p<<' ';
  }
  cout<<i;
}

```

#### c)

```

1   var p,i,x:integer;
2   begin
3   i:=0; p:=0; read(x);
4   while x>0 do begin
5     inc(i); p:=p+x;
6     read(x);write(p/i:0:2);
7   end;
8   end.
9

```

```

#include <iostream.h>
int i,x; float p;
void main() {
  i=0; p=0; cin>>x;
  while (x) {
    i++; p+=x;
    cin>>x; cout<<p/i<<' ';
  }
}

```

#### Soluție test 12:

- a) 22222;  
 b) n=5 și numerele 10, 20, 300, 4000, 5000;

#### c)

```

1   var n,i,nr,s,x:integer;
2   begin
3   read(n); s:=0;
4   for i:=1 to n do begin
5     nr:=1; read(x);
6     while x>10 do begin
7       x:=x div 10;
8       nr:=nr*10;
9     end;
10    s:=s+x*nr;
11  end;
12  writeln(s);
13  end.

```

```

#include <iostream.h>
int i,x,n,nr,s;
void main() {
  cin>>n; s=0;
  for(i=1;i<=n;i++) {
    nr=1; cin>>x;
    while (x>10) {
      nr*=10; x/=10;
    }
    s+=nr*x;
  }
  cout<<s;
}

```

#### Soluție test 13:

- a) \*  
 \*\*  
 \*\*\*  
 \*\*\*\*

```

1 b)
2 integ n,i,j;
3 citește n;
4 pentru i←n,1,-1 executa
5     pentru j←1, n-i executa
6         scrie ' ';
7     pentru j←n-i+1, n executa
8         scrie ' ';
9     scrie salt la linie noua
10
11
12

```

```

d)
1 integ n,i,j;
2 citește n;
3 pentru i←1, n executa
4     pentru j←n-i+1, n executa
5         scrie ' ';
6     pentru j←1, n-i executa
7         scrie ' ';
8     scrie salt la linie noua
9
10
11

```

```

c)
1 var n,i,j:integer;
2 begin
3     read(n);
4     for i:=1 to n do begin
5         for j:=1 to n-i do write('
6             ');
7         for j:=n-i+1 to n do
8             write('*');
9     writeln;
10 end;
11 end.

```

```

#include <iostream.h>
1 int n,i,j;
2 void main(){
3     cin>>n;
4     for (i=1;i<=n;i++){
5         for (j=1;j<=n-i;j++)
6             cout<<" ";
7         for (j=n-i+1;j<=n;j++)
8             cout<<"*";
9         cout<<endl;
10    }
11 }

```

#### Soluție test 14:

a) 0.42; b) n=1;

```

c)
1 var x,nr,n,j:integer;
2 ok:boolean;
3 begin
4     readln(n); nr:=0; x:=n;
5     repeat
6         nr:=nr*10+n mod 10;
7         n:=n div 10;
8     until n=0;
9     ok:=true;
10    for j:=2 to trunc(sqrt(nr)) do
11        if nr mod j=0 then
12            ok:=false;
13        if ok then writeln(nr/x:0:2)
14        else writeln(x/nr:0:2);
15 end.

```

```

#include <stdio.h>
#include <math.h>
1 void main() {
2     int nr,n,j,ok; float x;
3     scanf("%d",&n); nr=0; x=n;
4     do {
5         nr=nr*10+n%10; n/=10;
6     } while (n);
7     ok=1;
8     for(j=2;j*j<=nr;j++)
9         if (nr%j==0) ok=0;
10    if (ok) printf("%.2f\n",nr/x);
11    else printf("%.2f\n",x/nr);
12 }

```

e) n=1 și n=11;

#### Soluție test 15:

a) 37 b) n=1;

55

73

c) n=14;

```

d)
1 var n,i,j:integer;
2 ok:boolean;
3 begin
4     read(n);
5     for i:=2 to n-2 do begin
6         ok:=true;
7         for j:=2 to trunc(sqrt(i)) do
8             if i mod j=0 then ok:=false;
9         for j:=2 to trunc(sqrt(n-i)) do
10            if (n-i) mod j=0 then
11                ok:=false;
12         if ok then
13             writeln(i, ' ',n-i);
14     end;
15 end.

```

```

#include <iostream.h>
#include <math.h>
1 void main(){
2     int n,i,j,ok; cin>>n;
3     for (i=2;i<=n-2;i++){
4         ok=1;
5         for (j=2;j*j<=i;j++)
6             if (i%j==0) ok=0;
7         for (j=2;j*j<=n-i;j++)
8             if ((n-i)%j==0) ok=0;
9         if (ok)
10            cout<<i<<" "<<n-i<<endl;
11    }
12 }

```

e) Instrucțiunea pentru  $i \leftarrow 2, n-2$  executa devine pentru  $i \leftarrow 2, [n/2]$  executa.

#### Soluție test 16:

a) 5 9; b) n=4 și 25, 35, 435, 15;

```

c)
1 var i,n,x,c,max1,max2:integer;
2 begin
3     read(n);
4     max1:=-1; max2:=-1;
5     for i:=1 to n do begin
6         read(x); c:=x mod 10;
7         if c>max1 then begin
8             max2:=max1;
9             max1:=c;
10        end
11        else
12            if (c>max2) and (c<=max1)
13                then max2:=c;
14        end;
15        writeln(max2, ' ',max1);
16    end.

```

```

#include <iostream.h>
1 void main(){
2     int i,n,x,c,max1,max2;
3     cin>>n;
4     max1=-1; max2=-1;
5     for(i=1;i<=n;i++){
6         cin>>x; c=x%10;
7         if (c>max1){
8             max2=max1;
9             max1=c;
10        }
11        else if (c>max2 && c!=max1)
12            max2=c;
13    }
14    cout<<max2<<" "<<max1;
15 }

```

```

d)
1 integ n,i,max,nr,x,c; citește n;
2 max←-1; nr←0;
3
4 pentru i←1, n executa
5     citește x; c←x mod 10;
6     dacă c>max atunci
7         max←c; nr←1;
8     altfel
9         dacă (c=max)
10            atunci nr←nr+1;
11
12
13 scrie max, nr;

```

### Soluție test 17

a) 25, 313, 3, 1, 502;

b) 11, 1, 91, 31;

c)

```
1 var i,x,n,c:integer;
2 begin
3   read(n);
4   for i:=1 to n do begin
5     read(x);
6     c:=0;
7     repeat
8       c:=c*10+x mod 10;
9       x:=x div 100;
10    until x=0;
11    writeln(c);
12  end;
13 end.
```

```
#include <iostream.h>
void main(){
  int i,x,n,c;
  cin>>n;
  for (i=1;i<=n;i++){
    do {
      c:=c*10+x%10;
      x=x/100;
    } while (x);
    cout<<c;
  }
}
```

d)

```
1 integ n,i,c,x;
2 citește n; i←1;
3 cat timp i<=n executa
4   citește x;
5   c←0;
6   cat timp x>0 executa
7     c←c*10+x mod 10;
8     x←[x/100]
9   scrie c; i←i+1;
10 stop.
```

### Secțiunea 1.2.4

1.

```
1 citește n; p ← 1;
2 pentru i ← 1, n executa
3   p ← p*2*i
4 scrie p
```

2.

```
1 citește n; s ← 0; nr ← 0;
2 pentru i ← 1, n executa
3   citește y;
4   ok ← True;
5   pentru j ← 2, y div 2 executa
6     dacă y mod j = 0 atunci ok ← False
7   scrie ok
8   //verific dacă numărul y este prim
9
```

```
10   dacă ok atunci
11     s ← s + y;
12     nr ← nr+1;
13 scrie s / nr
```

3.

```
1 citește n; p ← 1;
2 pentru i ← 1, n executa
3   citește y;
4   x ← n;
5   z ← y;
6   cat timp x<>z executa
7     dacă x>z atunci x ← x-z
8     altfel z ← z-x
9   //determin cmmdc dintre y și n
10  dacă x = 1 atunci p ← p * y;
11 scrie p
```

4.

```
1 citește n, p d ← n div 2; nr ← 0;
2 cat timp (d > 1) and (nr < p) executa
3   dacă n mod d = 0 atunci
4     scrie d; nr ← nr + 1;
5   d ← d - 1; //se parcurg în sens invers divizorii lui n
6 scrie p
```

5.

```
1 citește a, b; s ← 0; nr ← 0;
2 pentru i ← a, b executa
3   x ← i; y ← 0;
4   cat timp x<>0 executa
5     y ← y * 10 + x mod 10
6     x ← x div 10
7   //determin inversul lui i în variabila y
8   dacă y = i atunci s ← s + i; nr ← nr + 1;
9 scrie s / nr
```

9.

```
1 {Fie min și max minimul și maximul celor 3 valori citite}
2 x ← min; p ← 1;
3 cat timp x<>0 executa
4   p ← p * 10 //calculăm 10^v unde v este numărul de cifre
5   x ← x div 10 //al valorii minime
6 scrie p * max + min
```

11.

```

1 citeste s;
2 pentru i ← 1, s div 2 executa
3   pentru j ← i + 1, s executa
4     daca (s-i-j>j) atunci
5       scrie i, j, s - i - j
6       //generez toate tripletele (i, j, k)
7       //de valori crescatoare de suma S
8

```

13.

```

1 citeste n, x, y ; nr ← 0;
2 pentru i ← 3, n executa
3   citeste z;
4   daca (x+y>z)and(x+z>y)and(y+z>x) atunci
5     nr ← nr + 1
6
7   x ← y; y ← z;
8
9 scrie nr

```

15.

```

1 citeste n, max1, max2 ; //citesc primele 2 valori din sir
2 daca max1 < max2 atunci max1 ← max2;
3
4 pentru i ← 3, n executa
5   citeste x;
6   daca (x > max1) atunci max2 ← max1; max1 ← x;
7   altfel daca (x > max2) atunci max2 ← x;
8
9   //max2, max1 valorile maxime din sir (max2 < max1)
10

```

16.

```

1 citeste n, max; //initializez maximul cu prima valoare din sir
2 nr ← 1; //numarul de aparitii al valorii maxime
3 pentru i ← 2, n executa
4   citeste x;
5   daca (x > max) atunci max ← x; nr ← 1;
6   altfel daca (x = max) atunci nr ← nr + 1;
7
8
9

```

19.

```

1 citeste n, a, b
2 pentru i ← 2, n executa
3   citeste x, y;
4   daca (x > b) atunci nr dij ← nr dij + 1; //disjuncte
5
6   daca (y < b) atunci nr inc ← nr inc + 1; //incluse
7
8

```

20.

```

1 citeste n; nr ← 0;
2 executa
3   n ← n + 1;
4   //Se verifica daca n este prim
5   daca (n este prim) atunci
6     scrie n; nr ← nr + 1;
7
8   pana_cand nr = 2

```

31.

```

1 citeste n;
2 x ← 1;
3 nr ← 0;
4 executa
5   daca (2*x+1)mod n = 0 atunci
6     scrie x, x + 1;
7     nr ← nr + 1;
8
9   x ← x + 1;
10  pana_cand nr = n

```

36.

```

1 citeste x;
2 executa
3   scrie x mod b;
4   x ← x div b;
5   pana_cand x = 0;

```

54.

```

1 citeste n; nr ← 0;
2 pentru i ← 1, n executa
3   x ← i;
4   cat_timp x mod 5 = 0 atunci
5     nr ← nr + 1; //determin nr. de factori de 5 din factorial
6     x ← x div 5;
7
8

```

57.

```

1 citeste a, b, x; nr ← 0;
2 cat_timp x <> 0 executa
3   y ← 0;
4   cat_timp x > 9 atunci
5     daca (x mod 10=b)and(x div 10 mod 10 =a)atunci
6       y ← 1;
7
8     x ← x div 10;
9
10  nr ← nr + y;
11 citeste x

```

58.

```

1 citeste x;
2 max ← 0;
3 p ← 10;
4 cat timp y <> 0 executa
5   daca (x div p) * (x mod p) > max atunci
6     max ← (x div p) * (x mod p);
7   p ← p * 10; y ← y div 10
8   ■
9   ■

```

60.

```

1 citeste x;           //determin cifra de control a unui nr x
2 cat timp x > 9 executa
3   s ← 0;
4   cat timp x > 0 executa
5     s ← s + x mod 10;
6     x ← x div 10;
7   ■
8   x ← s;
9   ■

```

### Secțiunea 1.3.2

1. Se determină numărul total de cercetași din zona Gălăciuc și numărul de cercetași din zona Soveja. Numărul de cercetași din fiecare detașament după reorganizare va fi cel mai mare divizor comun al celor două numere.

2. Se determină cel mai mare pătrat perfect mai mic strict ca  $n$  și apoi se afișează matricea.

3. Șirul este format din numere prime care au ultima cifră egală cu 7.

```

1 intreg n, i, j; logic ok;
2 citeste n;
3 i ← 3;
4 cat timp n > 0 executa
5   ok ← true;
6   pentru j ← 2, [√i] executa
7     daca i mod j = 0 atunci ok ← false;
8   ■
9   daca (ok=true) and (i mod 10 ≠ 7) atunci
10    ok ← false;
11  ■
12  daca ok=true atunci n ← n-1;
13  ■
14  i ← i+2;
15  ■
16 scrie i-2; stop.

```

4. Răspunsul este  $5*n$ .

5. Se determină valorile pantă în timpul citirii.

```

1 intreg n, i, nr, c1, c2, x, y; logic ok;
2 citeste n;
3 x ← 0; min ← -108; max ← -108;
4 pentru i ← 1, n executa
5   citeste nr;
6   ok ← true; y ← nr;
7   c1 ← nr mod 10;
8   c2 ← (nr div 10) mod 10;
9   nr ← nr div 100;
10  cat timp nr > 0 executa
11    daca ((c1 < c2) and (c2 > nr mod 10)) or
12      ((c1 > c2) and (c2 < nr mod 10)) atunci ok ← false;
13    c2 ← nr mod 10;
14    nr ← nr div 10;
15  ■
16  daca ok=true atunci
17    x ← x+1;
18    daca min > y atunci min ← y;
19    daca max < y atunci max ← y;
20  ■
21  ■
22 scrie "Numarul de valori-panta: ", x;
23 scrie "Cea mai mica valoare-panta: ", min;
24 scrie "Cea mai mare valoare-panta: ", max;
25 stop.

```

6. Se determină răspunsul avansând an cu an.

7. Pentru fiecare număr de la 1 la  $n$  se determină exponentul lui  $k$  în descompunerea lui în factori primi.

```

1 intreg n, k, nr, exp;
2 citeste n, k;
3 exp ← 0;
4 pentru i ← 1, n executa
5   citeste nr;
6   cat timp nr mod k = 0 executa
7     exp ← exp + 1;
8     nr ← nr div k;
9   ■
10  ■
11 scrie exp;
12 stop.

```

8. Inițial Luni, nasul lui Pinocchio măsoara  $n$  centimetri, în fiecare zi de Luni, Marți, Miercuri, Joi și Vineri nasul crește cu câte  $p$  centimetri iar Sâmbăta și Duminica scade cu câte 1 cm.

Pentru a determina lungimea nasului după  $k$  zile, va trebui să transformăm cele  $k$  zile în săptămâni și zile:  $k$  zile  $\Rightarrow (k_s = k \text{ div } 7)$  săptămâni +  $(k_z = k \text{ mod } 7)$  zile. Într-o săptămână nasul crește cu  $(5*p-2)$  cm.

Pentru cele  $(k \bmod 7)$  zile rămase vom determina lungimea nasului lui Pinocchio astfel: dacă  $kz=6$  este vorba de ziua de Sâmbătă și  $L=L+5*kz-1$ , altfel e vorba de o zi lucrătoare și  $L=L+p*kz$ .

9. Din numerele naturale de la 1 la  $n$  câte sunt divizibile cu  $p$  (multipli lui  $p$ ), atâtea scânduri vor fi vopsite cu roșu, câți multipli de  $q$  există, atâtea scânduri vor fi vopsite în albastru. Din cele  $n$  scânduri vor fi vopsite cu ambele culori atâtea, câte numere există care sunt și multipli și ai lui  $p$  și ai lui  $q$ , adică multipli ai lui c.m.m.m.c.  $(p, q)$ .

10. În prima zi cangurul sare 7 metri. În ziua a doua cangurul sare în plus față de prima zi, de 10 ori mai mult, adică  $7+7*10=77$  metri. În a treia zi cangurul sare în plus față de prima zi, de 10 ori mai mult decât în a doua, adică  $7+77*10=777$  metri. Și așa mai departe. Deci, în  $n$  zile cangurul va sări  $7+77+777+7777+\dots$ .

11. Cerința a) se rezolvă determinând cifrele fiecărui număr. Pentru cerința b) se iau toate cifrele de la 9 la 0 și se vede de câte ori apare fiecare cifră atât în  $a$  cât și în  $b$ . Apoi se afișează această cifră de câte ori se găsește în  $a$  și  $b$ .

12. Se determină cifrele fiecărui număr pentru rezolvarea cerințelor.

13. Rezolvarea se bazează pe observația că numărul de rațe de pe fiecare rând respectă termenii șirului lui Fibonacci: 1 1 2 3 5 8 13, mai puțin primul termen, care lipsește.

```

1  intreg ka, kb, a, b, c, nr;
2  citește ka, kb;
3  a ← 0;
4  b ← 1;
5  cat timp (ka>0) and (kb>0) executa
6      c ← a + b;
7      nr ← nr + 1;
8      ka ← ka - (c+1) div 2;
9      kb ← kb - c div 2;
10     a ← b; b ← c;
11
12 ka ← ka + (b+1) div 2;
13 kb ← kb + b div 2;
14 scrie nr - 1, ka, kb;
15 stop.

```

14. Se determină dintr-o singură parcurgere lungimea celei mai lungi porțiuni continue parcurse fără să găfăie.

```

1  intreg n, x, y, z, d, max;
2  citește n;
3  x ← 0;
4  d ← 0;

```

```

5  max ← 0;
6  pentru i ← 1, n executa
7      citește y, z;
8      dacă x < y atunci
9          dacă max < d atunci max ← d;
10         d ← 0;
11
12      altfel
13          d ← d + z;
14
15      x ← y;
16
17 dacă max < d atunci max ← d;
18 scrie max
19 stop.

```

15. Se deformează triunghiul în unul dreptunghic isoscel. Cele două diagonale vor corespunde liniei și coloanei pe care se află numărul  $n$  în triunghiul deformat.

```

1  intreg n, l;
2  citește n;
3  l ← 1;
4  cat timp l < n executa
5      n ← n - l;
6      l ← l + 1;
7
8  scrie "A", l-n+1, " B", n;
9  stop.

```

16. Se citesc succesiv numere scrise pe stâlpi contorizând și numărul de becuri albe, respectiv galbene, depistate până în acel moment. Pe baza acestor informații se determină culoarea becului curent.

```

1  intreg n, i, nr, a, g;
2  citește n;
3  a ← 0; g ← 0;
4  pentru i ← 1, n executa
5      citește nr;
6      dacă i mod 2=0 atunci
7          dacă nr=g atunci
8              scrie "Becul ", i, ": alb";
9              a ← a + 1;
10
11          altfel
12              scrie "Becul ", i, ": galben";
13              g ← g + 1;
14
15      altfel
16          dacă nr=a atunci
17              scrie "Becul ", i, ": alb";
18              a ← a + 1;
19
20

```



```

21 | altfel
22 |   scrie "Becul ", i, ": galben";
23 |   g ← g + 1;
24 |
25 |
26 |
27 | stop.

```

17. În prima etapă se transformă fracția în una ireductibilă. Pentru a determina nivelul s-ar putea efectua scăderi repetate simulând operația inversă construcției fracțiilor pe niveluri, dar această abordare este prea înceată; se folosesc împărțiri în locul scăderilor.

```

1 | intreg m, n, a, b, niv, t;
2 | citește m, n;
3 | niv ← 0;
4 | a ← m;
5 | b ← n;
6 | cat timp b > 0 executa
7 |   t ← a;
8 |   a ← b;
9 |   b ← t mod b;
10 |
11 | m ← m div a; n ← n div a;
12 | cat timp (m > 1) and (n > 1) executa
13 |   daca m > n atunci
14 |     niv ← niv + m div n;
15 |     m ← m mod n;
16 |   altfel
17 |     niv ← niv + n div m;
18 |     n ← n mod m;
19 |
20 |
21 |
22 | scrie niv+m+n-1;
23 | stop.

```

18. Se distribuie numere consecutive pe diagonale, plasând pe orice diagonală elemente cu aceeași valoare (se consideră ca toate diagonalele sunt formate din  $n$  elemente, se lucrează cu indici  $\text{mod } n$ ). Pentru a completa până când suma elementelor este fix  $S$ , se adaugă câte o unitate elementelor tabloului pornind de la cea mai mare valoare, pe toată diagonală sa, și continuându-se cu valorile următoare.

19. Pentru a răspunde la cerințe se determină numerele prime din șir.

20. Fie  $\phi(x)$  = câte numere mai mici ca  $x$  sunt prime cu  $x$ . Rezultatul va fi  $1 + (\phi(2) + \dots + \phi(n))$ . Pentru a calcula  $\phi(x)$  se folosește formula:

$$\phi(x) = x \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots \left(1 - \frac{1}{p_k}\right)$$

Unde  $p_1, p_2, \dots, p_k$  sunt factorii primi din descompunerea lui  $x$ .

```

1 | intreg n, i, j, nr, phi, r;
2 | citește n;
3 | r ← 1;
4 | pentru i ← 2, n executa
5 |   nr ← i; phi ← nr;
6 |   pentru j ← 2, √nr executa
7 |     daca nr mod j = 0 atunci
8 |       phi ← (phi * (j - 1)) div j;
9 |
10 |   cat timp nr mod j = 0 executa
11 |     nr ← nr div j;
12 |
13 |   daca nr > 1 atunci
14 |     phi ← (phi * (nr - 1)) div nr;
15 |
16 |   r ← r + 2 * phi;
17 |
18 |
19 | scrie r;
20 | stop.

```

21. Se încearcă toate valorile posibile pentru  $k$  începând cu 1 până când se găsește cea corespunzătoare.

```

1 | intreg n, k, f;
2 | citește n;
3 | k ← 1; f ← 1;
4 | cat timp f < n executa
5 |   k ← k + 1;
6 |   f ← f * k;
7 |
8 | scrie k;

```

22. Se reprezintă numărul în fiecare bază și se determină suma cifrelor.

23. Din  $a^2 + b^2 = c^2$  se determină  $a^2 = c^2 - b^2 \Leftrightarrow a^2 = (c-b)(c+b)$ . Se determină toți divizorii  $d$  ai lui  $a^2$  și din relațiile  $c-b=d$  și  $c+b=a^2/d$  se determină  $c$  și  $b$ .

```

1 | intreg a, b, c, d;
2 | citește a;
3 | pentru d ← 1, a-1 executa
4 |   daca ((a*a+d*d) mod (2*d) = 0) and ((a*a-d*d) mod (2*d) = 0) atunci
5 |     b ← (a*a+d*d) div (2*d);
6 |     c ← (a*a-d*d) div (2*d);
7 |     scrie b, c;
8 |
9 |

```

24. Deoarece  $R$  are limita mică (30), se verifică dacă toate punctele  $(x, y, z)$  cu  $x, y, z \leq R$  sunt în sferă.

```

1 | intreg x, y, z, r, nr;
2 | citește r;
3 | nr ← 0;

```

```

4 pentru x ← 1, r executa
5   pentru y ← 1, r executa
6     pentru z ← 1, r executa
7       daca x*x+y*y+z*z ≤ r atunci
8         nr ← nr+1;
9       ■
10    ■
11  ■
12 ■
13 scrie nr;
14 stop

```

25. Dacă  $n$  este divizibil cu 3 rezultatul va fi  $3*3*...*3$ , dacă  $n$  are restul 2 la împărțirea cu 3 rezultatul va fi  $3*3*...*3*2$ , iar dacă are restul 1, rezultatul va fi  $3*3*...*3*4$ .

```

1  intreg n, y, z, r, nr;
2  citește n; nr ← 0;
3  daca n mod 3=0 executa
4    r ← 1;
5    pentru i ← 1, n div 3 executa
6      r ← r * 3;
7    ■
8    scrie r;
9  ■
10  daca n mod 3=1 executa
11    r ← 4;
12    pentru i ← 1, (n-4) div 3 executa
13      r ← r * 3;
14    ■
15    scrie r;
16  ■
17  daca n mod 3=2 executa
18    r ← 2;
19    pentru i ← 1, (n-2) div 3 executa
20      r ← r * 3;
21    ■
22    scrie r;
23  ■
24  stop

```

26. Pentru orice grup care trebuie format din  $n-2$  persoane (conform cerințelor problemei) trebuie să existe cel puțin un lacăt pe care nu-l va deschide nimeni din grup. Rezultă că celelalte două persoane rămase în afara grupului dețin fiecare cheia lacătului respectiv.

Astfel, atunci când una din cele două persoane intră în grupul de  $n-2$  persoane, lacătul va putea fi deschis. Se deduce că la fiecare lacăt trebuie să existe exact două chei.

Tot de aici se deduce că numărul de lacăte este egal cu numărul grupurilor distincte formate din  $n-2$  persoane, adică  $n*(n-1)/2$ . Atunci numărul total de chei va fi:  $n*(n-1)$ .

```

1  intreg n, i, j, nr;
2  citește n;
3  nr ← 0;
4  scrie "lmin=", n*(n-1) div 2, " chei=", n*(n-1);
5  pentru i ← 1, n-1 executa
6    pentru j ← i+1, n executa
7      scrie "lacat ", nr, ":", i, " ", j;
8      nr ← nr + 1;
9    ■
10  ■
11  stop.

```

27. Rezultatul cerut este  $(A+1)*(B+1)*(A+B+2)/2$ .

28. Numărul de pe linia  $L$  și coloana  $C$  va fi  $(L-1) \text{ xor } (C-1)$ .

29. Se folosește căutarea binară pentru determinarea rezultatului. Pentru a determina câte cifre de 0 are la sfârșit produsul  $1*2*...*nr$  se determină exponentul lui 5 în descompunerea produsului folosind formula:

$$\left\lfloor \frac{nr}{5} \right\rfloor + \left\lfloor \frac{nr}{5^2} \right\rfloor + \left\lfloor \frac{nr}{5^3} \right\rfloor + \left\lfloor \frac{nr}{5^4} \right\rfloor + \dots + 0$$

30. Se face elementul 3 invizibil în șirul mutărilor și se scade cu o unitate fiecare număr. Se obține astfel secvența inițială de mutări, doar că fiecare al 3-lea pas se repetă. Eliminând pașii care se repetă sunt necesare  $f(n-1)$  mutări pentru a aduce  $n-1$  pe prima poziție (care era  $n$  în prima secvență). Fiindcă s-au eliminat exact  $(f(n-1)+1) \text{ div } 2$  pași, se deduce relația  $f(n)=f(n-1)+(f(n-1)+1) \text{ div } 2$ .

```

1  intreg n, i, nr;
2  citește n;
3  nr ← 1;
4  pentru i ← 4, n executa
5    nr ← nr + (nr+1) div 2;
6  ■
7  scrie nr;
8  stop.

```

### Teste cu alegere multiplă și duală din capitolul 2

2.1.1	2.1.1	2.2.1	2.2.1	2.3.1
1. a), d)	9. b)	1. a), c)	11. b), d)	1. d)
2. c)	10. a), c)	2. a), b)	12. d)	2. b)
3. b), c)	11. d)	3. b), c)	13. c)	3. c)
4. b)	12. b)	4. d), e)	14. c)	4. b), d)
5. b), c)	13. b), d)	5. d)	15. c)	5. d)
6. d)	14. a), e)	6. c)	16. b)	6. a)
7. a)	15. b)	7. b), d)	17. b)	7. b)
8. d)		8. d)	18. c)	8. c)
		9. c)	19. b)	9. b)
		d)		

## Sectiunea 2.1.2

### Soluție test 1

- a) 0 1 2 4 3 6;    b) Orice șir de valori ordonate crescător;  
c) n=6 și șirul 0 2 1 4 3 5;  
d)

```
1 var a : array[1..100] of byte;
2 i, n, x: integer;
3 begin
4   read(n);
5   for i:=1 to n do read(a[i]);
6   for i:=1 to n-1 do
7     if a[i]>a[i+1] then begin
8       x:=a[i]; a[i]:=a[i+1]; a[i+1]:=x;
9     end;
10  for i:=1 to n do write(a[i]);
11 end.
12
```

```
#include <iostream.h>
unsigned char a[100];
int i,n,x;
void main() {
  cin>>n;
  for (i=0;i<n;i++) cin>>a[i];
  for (i=0;i+1<n;i++)
    if (a[i]>a[i+1]) {
      x=a[i]; a[i]=a[i+1]; a[i+1]=x;
    }
  for (i=0;i<n;i++) cout<<a[i];
}
```

### Soluție test 2

- a) 22, 32, 10, 26, 16;    b) Orice șir cu elemente mai mari strict decât 9;  
c) Instrucțiunea de la linia 7 devine:

daca a[i] < 10 atunci ...

```
1 var a : array[1..100] of byte;
2 i, n: integer;
3 begin
4   read(n);
5   for i:=1 to n do read(a[i]);
6   for i:=2 to n-1 do
7     if a[i] div 10 = 0 then begin
8       a[i] := a[i-1] + a[i+1];
9     end;
10  for i:=1 to n do write(a[i]);
11 end.
```

```
#include <iostream.h>
void main() {
  int i,n, a[100];
  cin>>n;
  for (i=0;i<n;i++) cin>>a[i];
  for (i=1;i+1<n;i++)
    if (a[i]/10==0)
      a[i]=a[i-1]+a[i+1];
  for (i=0;i<n;i++) cout<<a[i];
}
```

### Soluție test 3

- a) 3, 4, 1, 2;  
b) Orice șir pentru care prima și ultima valoare sunt diferite;  
c) Orice șir de valori egale;  
d)

```
1 var a : array[1..100] of byte;
2 i, n, j, k: integer;
3 begin
4   read(n);
5   for i:=1 to n do read(a[i]);
6   i:=1; j:=n;
7   while (a[i]=a[j]) and (i<=j) do
```

```
#include <iostream.h>
unsigned char a[100];
int i,n,j,k;
void main() {
  cin>>n;
  for (i=0;i<n;i++) cin>>a[i];
  i=0; j=n-1;
```

```
8 begin
9   inc(i); dec(j);
10  end;
11  for k:=i to j do write(a[k]);
12 end.
```

```
while (a[i]==a[j] && i<=j) {
  i++; j--;
}
for (k=i;k<=j;k++) cout<<a[k];
}
```

### Soluție test 4

- a) 1 4 3 0 2;  
b) Orice număr palindrom (egal cu numărul citit de la dreapta la stânga);

```
1 var a : array[1..100] of byte;
2 i, n, x: integer;
3 begin
4   readln(x); n:=0;
5   while x<>0 do begin
6     inc(n); a[n]:= x mod 10;
7     x:=x div 10;
8   end;
9   for i:=1 to n do write(a[i]);
10 end.
```

```
#include <iostream.h>
int a[100];
int i,n,x;
void main() {
  cin>>x; n=0;
  while (x!=0) {
    a[n++]=x%10;
    x/=10;
  }
  for (i=0;i<n;i++) cout<<a[i];
}
```

### Soluție test 5

- a) NU;  
b) Elementele situate pe poziții consecutive să fie de semne contrare;  
c) Orice șir de numere de același semn;

```
1 var a: array[1..100] of integer;
2 i, n, nr: integer;
3 begin
4   read(n); nr:=0;
5   for i:=1 to n do read(a[i]);
6   for i:=1 to n-1 do
7     if a[i]*a[i+1]<0 then inc(nr);
8   if nr=0 then write('DA')
9   else write('NU')
10 end.
```

```
#include <iostream.h>
int i,n,nr, a[100];
void main() {
  cin>>n; nr=0;
  for (i=0;i<n;i++) cin>>a[i];
  for (i=0;i+1<n;i++)
    if (a[i]*a[i+1]<0) nr++;
  if (nr==0) cout<<"DA";
  else cout<<"NU";
}
```

### Soluție test 6

- a) 2, 3, 6, 1 respectiv 2, -1;  
b) Orice vector în care nu există nici un element egal cu suma vecinilor săi;  
d) Nu există. Primul și ultimul element nu pot fi șterse;  
e)

```
1 var a: array[1..100] of integer;
2 n,j,i: integer;
```

```
#include <iostream.h>
int a[100],n,i,j;
```

```

3 begin
4   readln(n);
5   for i:=1 to n do read(a[i]);
6   i:=2;
7   while i<n do
8     if a[i]=a[i-1]+a[i+1] then
9       begin
10        for j:=i to n do a[j]:=a[j+1];
11        dec(n);
12      end
13    else inc(i);
14    for i:=1 to n do write(a[i], ' ');
15  end.

void main() {
  cin>>n;
  for (i=0;i<n;i++) cin>>a[i];
  i=1;
  while (i+1<n)
    if (a[i]==a[i-1]+a[i+1]) {
      for (j=i;j<n;j++) a[j]=a[j+1];
      n--;
    }
    else i++;
  for (i=0;i<n;i++)
    cout<<a[i]<<' ';
}

```

#### Soluție test 7

a)2; b)Orice șir de  $n$  valori pentru care cifra care apare de cele mai multe ori în scrierea lor este 0;

```

e)
1 var nr:array[0..100]of integer; #include <iostream.h>
2 n,c,max,i,j:integer; int nr[101],n,c,max,i,j;
3 begin void main() {
4   readln(n); cin>>n; max=0;
5   max:=0; for (i=0;i<n;i++) {
6   for i:=1 to n do begin cin>>j;
7     readln(j); while (j!=0) {
8     while j<>0 do begin nr[j%10]++;
9       inc(nr[j mod 10]); if (max<nr[j%10]) {
10      if max<nr[j mod 10] then max=nr[j%10];
11      begin c=j%10;
12        max:=nr[j mod 10]; }
13        c:=j mod 10; }
14      end; j/=10;
15      j:=j div 10; }
16    end; cout<<c<<endl;
17  end; }
18  writeln(c); end.

```

#### Soluție test 8

a) 2 2 34 \* 5 6 78 \* 8 3; b) Orice șir crescător de  $n$  valori;  
c) Separați prin caracterul '\*' secvențele monotone crescătoare în care poate fi împărțit vectorul  $A$  de lungime  $n$ ;  
d)  $n-1$ . În cazul unui vector cu valori ordonate descrescătoare;

```

e)
1 var a:array[1..100]of integer; #include <iostream.h>
2 n,max,i:integer; int a[100],n,max,i;
3 begin void main() {
4   readln(n); cin>>n;
5   max:=1; max=1;
6   for i:=1 to n do read(a[i]); for (i=0;i<n;i++) cin>>a[i];

```

```

7   write(a[1], ' '); cout<<a[0]<<' ';
8   for i:=2 to n do for (i=1;i<n;i++)
9     if a[i]<a[i-1] then begin if (a[i]<a[i-1]) {
10      inc(max); max++;
11      write('* ',a[i], ' '); cout<<"* "<<a[i]<<" ";
12    end }
13    else write(a[i], ' '); else cout<<a[i]<<" ";
14    writeln(max); cout<<max<<endl;
15  end.

```

#### Soluție test 9

a) 2 respectiv 0; b)  $m$  reprezintă mijlocul secvenței din vector situate între indicii  $i$ ,  $j$ . Valoarea variabilei  $p$  reprezintă poziția în șir pe care s-a regăsit valoarea  $x$ ;  
d) 10 operații ( $1024=2^{10}$ );

```

e)
1 var a:array[1..100]of integer; #include <iostream.h>
2 n,i,j,x,p,m:integer; int a[100],n,i,j,x,p,m;
3 begin void main() {
4   readln(n,x); cin>>n>>x;
5   for i:=1 to n do read(a[i]); for (i=0;i<n;i++) cin>>a[i];
6   p:=0; p=-1;
7   i:=1; i=0;
8   j:=n; j=n-1;
9   while (i<=j) and (p=0) do begin while (i<=j && p==1) {
10    m:=(i+j) div 2; m=(i+j)/2;
11    if a[m]=x then p:=m; if (a[m]==x) p=m;
12    else else
13      if x<a[m] then j:=m-1; if (x<a[m]) j:=m-1;
14      else i:=m+1; else i:=m+1;
15    end; }
16    writeln(p); cout<<p+1<<endl;
17  end.

```

#### Soluție test 10

a) 9, 4, 2 respectiv 23; b) Nu există;  
c)  $nr$  reprezintă numărul maxim de subșiruri crescătoare în care poate fi partiționat un șir de  $n$  valori;  
d) Șirul de valori trebuie să fie ordonat strict descrescător;

```

e)
1 var a:array[1..100]of integer; #include <iostream.h>
2 n,k,i,x,nr:integer; int a[100],n,k,i,x,nr;
3 begin void main() {
4   read(n,a[1]); nr:=1; cin>>n>>a[0];nr:=1;
5   for i:=2 to n do begin for (i=1;i<n;i++) {
6     read(x); k:=1; cin>>x; k=0;
7     while (x<=a[k]) and (k<=nr) do while (x<=a[k] && k<nr) k++;
8       inc(k); if (k==nr) nr++;
9       if k=nr+1 then inc(nr); a[k]=x;
10      a[k]:=x; }
11    end; for (i=0;i<nr;i++) cout<<a[i];
12  for i:=1 to nr do write(a[i]); }
13  end.

```

## Secțiunea 2.1.4

```

2.
1 max ← a[1]; nr ← 1;
2 pentru i ← 2, n executa
3   dacă a[i] > max atunci
4     max ← a[i]; nr ← 1;           // se reinitializeaza max
5   altfel
6     dacă a[i] = max atunci nr ← nr + 1;
7   ■
8   ■
9   ■

```

```

6.
1 ns ← 0; nd ← 0;
2 pentru i ← 1, n executa
3   ns ← ns * 10 + a[i];           //numarul de la stanga la dreapta
4   nd ← nd * 10 + a[n-i+1]       //numarul de la dreapta la stanga
5   ■

```

```

10.
1 pentru i ← 1, n-1 executa
2   pentru j ← i+1, n executa
3     dacă (a[i] < a[j]) and (a[i]*a[j] < 0) atunci
4       a[i] ↔ a[j]               // se interschimba elemente nenule
5     ■
6   ■
7   ■

```

```

11.
1 x ← a[1]
2 cat timp i < n+1 executa
3   dacă a[i] = x atunci
4     pentru j ← i, n-1 executa
5       a[j] ← a[j+1]             // se sterge elementul a[i]
6     ■
7     n ← n - 1                   //se micsoreaza numarul de elemente
8     altfel i ← i + 1           //se continua parcurgerea
9   ■
10  ■

```

```

12.
1 Pasul 1: Ordonare vector A
2 i ← 1;
3 nr ← 0;
4 cat timp i < n+1 executa
5   x ← i
6   cat timp (a[i] = a[x]) and (i < n) executa
7     i ← i + 1                   // se parcurge vectorul cat timp
8     ■                           // elementele sunt egale cu a[x]
9   dacă (i - x) > nr atunci
10    nr ← i - x                  //actualizez nr maxim de aparitii
11    v ← a[x]
12  ■
13  ■                               //se continua parcurgerea
14  scrie v, nr

```

```

14.
1 p ← 1; nr ← 0; max ← 0;
2 pentru i ← 1, n executa
3   dacă a[i] > 0 atunci           //primului element pozitiv din
4     dacă nr = 0 atunci p ← i    // secventa curenta i se retine
5     ■                           //in variabila p pozitia in sir
6     inc(nr)
7   ■
8   dacă nr > max atunci           //lungimea secv. curente este >
9     max ← nr; pl ← p;           //actualizez lung. maxima
10  ■
11  dacă a[i] < 0 atunci nr ← 0    //la un element negativ resetez
12  ■                               //lungimea secventei
13  ■
14  pentru i ← pl, pl+max-1 executa scrie a[i]
15  ■

```

```

16.
1 m ← 0;
2 pentru i ← 1, n executa         //construim vectorul B ce cuprinde
3   ok ← false;                   // elementele din A fara repetitii
4   pentru j ← 1, m executa
5     dacă a[i] = b[j] atunci ok ← true;
6   ■
7   ■
8   dacă not ok atunci           // plasam un nou element in B
9     m ← m + 1; b[m] ← a[i]
10  ■
11  ■
12  pentru i ← 1, m executa
13    nr ← 0;                       // determin numarul de aparitii ptr b[i]
14    pentru j ← 1, n executa
15      dacă b[i] = a[j] atunci nr ← nr + 1;
16    ■
17    scrie b[i], nr
18  ■
19  ■

```

```

18.
1 pentru p ← 1, n executa         //se genereaza cele n permutari
2   a[n+1] ← a[1];                //a[1] se plaseaza pe pozitia n+1
3   pentru j ← 1, n executa
4     a[j] ← a[j+1]               // se sterge primul element
5   ■
6   pentru j ← 1, n executa
7     scrie a[j]
8   ■
9   ■

```

```

23.
1 min ← a[1]; max ← a[1];
2 pl ← 1; p2 ← 1;
3 pentru i ← 1, n executa
4   dacă a[i] > max atunci max ← a[i]; pl ← i;
5   ■                               //pl = pozitia maximului

```

```

30.  1  pentru i ← 1, n executa citeste a[i]
      2  ■
      3  pentru i ← 1, n executa
      4  x ← a[i]; y ← (i+a[i]) mod n; // se interchimba elementul

```

```

36. 1 i ← 1;
      2 cat timp i < n executa
      3   [daca a[i]*a[i+1] < 0 atunci //a[i] si a[i+1] au semne contrare
      4     x ← |a[i]|;
      5     y ← |a[i+1]|;
      6     [cat timp y < 0 executa // se inmulteste x 10 (nr.cifre y)
      7       x ← x*10; y ← y div 10;
      8     ]
      9     pentru j ← n, i+1, -1 executa a[j+1] ← a[j];
     10     [ //se deplaseaza elemente cu 1 pozitie
     11       a[i+1] ← x + |a[i+1]| //se insereaza valoarea ceruta
     12       i ← i + 2;
     13       n ← n + 1;
     14     ] altfel i ← i + 1; //se continua parcurgerea
     15   ]
     16

```

```

38.
1 citeste n, a[1]; nr ← 1; //a[i] este ultimul element dintr-un
2 pentru i ← 2, n executa //subsir cu elemente consecutive
3 citeste x; k ← 1; //caut subsirul unde poate fi asezat x
4 cat timp (x*a[k]+1) and (k<=nr)
5     executa k ← k + 1;
6
7 daca k=nr + 1 atunci //daca nu exista il plasez pe o noua
8     nr ← nr + 1; //pozitie in vectorul a
9
10 a[k] ← x;
11
12 scrie nr; //afisez nr de subsiruri cu elemente consecutive

```

```

40.
1 pentru i ← 0, n-1 executa //r[i]=x codifica secventa
2 r[i] ← 0; // a[1]+...+a[x] are restul mod n egal cu i
3
4 s ← 0
5 pentru i ← 1, n executa
6 s ← s + a[i]; //suma primelor i elemente
7 daca s mod n = 0 atunci //are restul mod n=0 => este solutie
8     p1 ← 1;
9     p2 ← i;
10    i ← n;
11 altfel
12     daca r[s mod n] ≠ 0 atunci //restul s mod n s-a mai obtinut
13         p1 ← r[s mod n] + 1; p2 ← i; i ← n;
14     altfel
15         r[s mod n] ← i; //se retine indicele i
16
17
18
19 pentru i ← p1, p2 executa scrie a[i];
20

```

```

40.
1 pentru i ← 0, n-1 executa //r[i]=x codifica secventa
2 r[i] ← 0; // a[1]+...+a[x] are restul mod n egal cu i
3
4 s ← 0
5 pentru i ← 1, n executa
6 s ← s + a[i]; //suma primelor i elemente
7 daca s mod n = 0 atunci //are restul mod n=0 => este solutie
8     p1 ← 1; p2 ← i; i ← n;
9 altfel
10     daca r[s mod n] ≠ 0 atunci //restul s mod n s-a mai obtinut
11         p1 ← r[s mod n] + 1; p2 ← i; i ← n;
12     altfel
13         r[s mod n] ← i; //se retine indicele i
14
15
16
17 pentru i ← p1, p2 executa scrie a[i];
18

```

```

43.
1 p ← 1; nn ← 0; //numarul de nr negative
2 pentru i ← 1, n executa
3     daca a[i] < 0 atunci nn ← nn + 1;
4
5     daca |a[i]| < |a[p]| then p:=i; //determin pozitia valorii
6     // minime in modul care va fi asezata pe pozitia n in A
7     //a[n] va fi eliminat
8 x ← a[p];
9 a[p] ← a[n];
10 a[n] ← x;
11 daca x < 0 atunci nn ← nn - 1;
12
13 Ordonam primele n-1 valori din vector
14 daca k mod 2 ≠ nn mod 2 atunci k ← k - 1;
15 //consider ca lui a[n] i se schimba semnul
16 pentru i ← 1, k executa a[i] ← (-1)*a[i];
17
18 pentru i ← 1, n-1 executa // calculeaza produsul, se afiseaza
19 p:=p*a[i]; scrie a[i] // valorile cu semn schimbat
20
21 scrie p

```

```

44.
1 a[0] ← 0; a[1] ← 1; //initializarea primilor termeni
2 pentru i ← 2, n executa
3     a[i] ← i + a[i div 2]; // relatia de recurenta
4
5 scrie a[n]

```

## Sectiunea 2.2.2

### Soluție test 1

- a) 3 11; b) Orice tablou bidimensional pentru care sumele pe linii ale cifrelor unităților formează un șir crescător;  
c) Orice tablou bidimensional cu elemente în mulțimea  $0...9$ ;

### Soluție test 2

- a) 2; b) Valoarea maximă poate fi  $n$ .  
c) Tabloul nu conține nici un element divizibil cu 3;  
d) Considerăm că variabila  $c$  va reține indicele coloanei ce urmează a fi afișată.  
Instrucțiunea de pe linia 16 devine:

```
daca x>max atunci
    max ← x; c ← j;
```

### Soluție test 3

- a) 3 2; b) Valoarea maximă a lui  $max$  poate fi  $m-1$ , în cazul în care există o linie cu toate elementele egale;  
c) Orice tablou în care elementul de pe prima coloană nu se mai regăsește în cadrul liniei sale;

### Soluție test 4

- a) Vor fi afișate pe linii valorile 1 2 3 4, 1 2 3 4, 1 2 3 4, 1 2 3 4;  
b) Elementele în cadrul unei linii sunt egale cu numărul liniei respective;  
c) Se pot folosi indicii folosiți la liniarizarea matricii:

```
pentru i ← 1, n executa
    pentru j ← 1, n executa
        a[i,j] ← 2*((i-1)*n+j);
```

### Soluție test 5

- a) Elementele tabloului pe linii vor fi: 1 2 3, 6 5 4, 7 8 9, 12 11 10;  
b) Elementele se vor completa după aceeași regulă, dar începând cu valoarea  $n*m$ ;

### Soluție test 6

- a) 1 1 1 0      b) Orice valori astfel încât  $m = n + 1$ ;  
    2 2 0 0  
    3 0 0 0  
    0 0 0 0

- c) Notăm cu  $x$  valoarea minimă dintre  $n$  și  $m$ . Numărul de elemente nule este:  
 $nr = x*(x+1) \div 2$  , dacă  $n \leq m$   
 $nr = x*(x+1) \div 2 + abs(n-m)*x$  , dacă  $n > m$

### Soluție test 7

- a) 15; c) Pe liniile 9 și 10 instrucțiunile devin:

```
pentru i ← (n+1) div 2, n executa
    pentru j ← n-i+1, i executa
        . . . . .
```

### Soluție test 8

- a) 1440; c) Pe liniile 9 și 10 instrucțiunile devin:

```
pentru j ← (n+1) div 2, n executa
    pentru i ← n-j+1, j executa
        . . . . .
```

### Soluție test 9

- a) 

1	2	3	4
5	6	7	16
8	9	15	14
10	13	12	11

 b)  $n=2$ ; c) Pe linia 5, instrucțiunea devine:  
pentru j ← 1, N - 1 executa...

### Soluție test 10

- a) 

1	2	3	4
2	1	2	3
3	2	1	2
4	3	2	1

 b)  $n=2$ ; c) Linia 5 devine:  $a[i,j] \leftarrow |i-j|$ ;

## Sectiunea 2.2.4

```
7.
1  pentru j ← 1, n executa
2     x ← a[1,j] // primul element dupa coloana j
3     pentru i ← 2, n executa
4         y ← a[i,j]
5         cat timp y <> 0 executa // calcul c.m.m.d.c. prin Euclid
6             r ← x mod y; x ← y; y ← r
7         scrie x //afisarea c.m.m.d.c pe coloana j
8
9
10
```

```
10.
1  pentru j ← 1, m executa
2     a[n+1,j] ← a[1,j] //se copiaza linia 1 pe linia n+1
3
4  pentru i ← 1, n executa // se sterge linia 1 prin
5     pentru j ← 1, m executa // deplasare liniilor 2..n+1
6         a[i,j] ← a[i+1,j]
7
8
```



```

12.
1  pentru i ← 1, n executa // a[i,0] retine suma pe linia i
2  a[i,0] ← 0; // a[0,i] retine suma pe coloana i
3  pentru j ← 1, n executa
4  a[i,0] ← a[i,0] + a[i,j]; a[0,j] ← a[0,j] + a[i,j]
5
6
7  pentru i ← 1, n executa //se identifica elementele a[i,j]
8  pentru j ← 1, n executa //ptr care a[i,0]=a[0,j]
9  daca a[i,0] = a[0,j] atunci scrie a[i,j]
10
11
12

```

```

13.
1  pentru i ← 1, n executa
2  min ← a[i,1]; ok ← true // determin minimul pe linia i
3  pentru j ← 1, n executa
4  daca a[i,j] < min atunci
5  min ← a[i,j]; c ← j; //se retine coloana minimului
6
7
8  pentru l ← 1, n executa // se verifica daca min este
9  daca a[l,c] > min atunci // maxim pe coloana c
10 ok ← False;
11
12
13  daca ok atunci scrie min //afisare punct "sa"
14
15

```

```

21.
1  nr ← 0;
2  pentru i ← 1, n executa
3  ok ← true // presupunem ca toate elementele sunt egale
4  pentru j ← 1, m-1 executa
5  daca a[i,j] <> a[i,j+1] atunci
6  ok ← false // s-a identificat pereche de elemente ≠
7
8
9  daca ok atunci nr ← nr + 1
10
11

```

```

24
1  t ← 0;
2  pentru j ← m, 1, -1 executa
3  a[0,j] ← t; //se aduna numerele dupa alg. aritmetic
4  pentru i ← 1, n executa //a[0,j] cifra ce apare pe
5  a[0,j] ← a[0,j] + a[i,j] // pozitia j in scrierea sumei
6
7
8  t ← a[0,j] div 10; //t = restul de transport la adunare
9  a[0,j] ← a[0,j] mod 10 //cifra ramasa pe pozitia j

```

```

10 a[0,0] ← t; //ultimul rest de transport posibil ≠ 0
11 pentru j ← 0, m executa
12 scrie a[0,j], ' '
13

```

```

28.
1  pentru i ← 1, n executa //se ordoneaza elementele pe linii
2  pentru j ← 1, m-1 executa
3  pentru k ← j + 1, m executa
4  daca a[i,j] > a[i,k] atunci a[i,j] ↔ a[i,k]
5
6
7
8  Pasul 2: Ordonare elemente tabloul pe coloane

```

```

30.
1  pentru i ← 1, n executa //se sterg elementele pe diagonala
2  pentru j ← i, n-1 executa //prin deplasarea spre stanga o
3  a[i,j] ← a[i,j+1] // pozitie a urmatoarelor
4
5
6  m ← n-1; //tabloul are n linii m coloane

```

```

31.
1  pentru i ← 1, n executa
2  pentru j ← 1, m executa
3  a[i,j] ← i + j - 1
4
5

```

```

35.
1  intreg a[100][100], p, i, j, n, m, k, p1, p2;
2  citeste n, m; p1 ← 0; p2 ← 0;
3  Citirea elementelor matricii A
4  pentru i ← 0, n+1 executa //se boordeaza matricea cu 1.
5  a[i,0] ← 1; a[i,m+1] ← 1;
6
7  pentru i ← 0, m+1 executa
8  a[0,i] ← 1; a[n+1,i] ← 1;
9
10 pentru i ← 1, n executa //se actualizeaza pas cu pas
11 pentru j ← 1, m executa //valorile lui p1 si p2
12 daca a[i-1,j]*a[i,j-1]*a[i+1,j]*a[i,j+1] > p1 atunci
13 p2 ← p1; x2 ← x1; y2 ← y1; x1 ← i; y1 ← j;
14 p1 ← a[i-1,j]*a[i,j-1]*a[i+1,j]*a[i,j+1];
15 altfel
16 daca p2 < a[i-1,j]*a[i,j-1]*a[i+1,j]*a[i,j+1] atunci
17 p2 ← a[i-1,j]*a[i,j-1]*a[i+1,j]*a[i,j+1]; x2 ← i; y2 ← j
18
19
20
21
22 scrie(p1, ' ', x1, ' ', y1, ' ', p2, ' ', x2, ' ', y2);

```

36.

```

1  intreg a[100][100], s, i, j, p, n, m, x, y; caracter c;
2  Citirea datelor de intrare
3  s ← a[x,y]; //calculez suma elementelor de pe traseu
4  pentru i ← 1, p executa
5  citeste c; //indica directia deplasarii
6  daca c='N' atunci x ← x - 1;
7  ■
8  daca c='S' atunci x ← x + 1;
9  ■
10  daca c='V' atunci y ← y - 1;
11  ■
12  daca c='E' atunci y ← y + 1;
13  ■
14  s ← s + a[x,y]; //elementul in care se ajunge este insumat
15  ■
16  scrie s;

```

38.

```

1  intreg a[100][100], max, s, i, j, p, n, m, x, y, xi, yi;
2  Citirea datelor de intrare
3  pentru i ← 1, n executa
4  pentru j ← 1, m executa
5  daca a[i,j]=0 atunci //se poate plasa regina in loc liber
6  x ← i; y ← j; s ← 0; //insumez elementele de pe diagonale
7  cat timp (x>0)and(y>0) executa
8  s ← s + a[x,y]; x ← x - 1; y ← y - 1;
9  ■
10  x ← i; y ← j; //sunt 4 directii care descriu diagonalele
11  cat timp (x>0)and(y<m+1) executa
12  s ← s + a[x,y]; x ← x - 1; y ← y + 1;
13  ■
14  x ← i; y ← j;
15  cat timp (x<n+1)and(y>0) executa
16  s ← s + a[x,y]; x ← x + 1; y ← y - 1;
17  ■
18  x ← i; y ← j;
19  cat timp (x<n+1)and(y<m+1) executa
20  s ← s + a[x,y]; x ← x + 1; y ← y + 1;
21  ■
22  daca s>max atunci max ← s; xi ← i; yi ← j;
23  ■ //se actualizeaza nr. max de pionii atacati
24  ■
25  ■
26  ■
27  scrie max, ' ', xi, ' ', yi;

```

### Sectiunea 2.4.2

1. Se calculează produsul (mod 10) al tuturor factorilor primi, la puterile la care apar, ignorând toți factorii de 5 și un număr de factori de 2 egal cu numărul factorilor de 5. Vezi problema 4 secțiunea 2.4.1.

```

1  intreg n, i, n2, n5, u, nr;
2  citeste n; u ← 1;
3  pentru i=2, n executa
4  nr ← i;
5  cat timp nr mod 2=0 executa
6  nr ← nr div 2;
7  n2 ← n2 + 1;
8  ■
9  cat timp nr mod 5=0 executa
10 nr ← nr div 5;
11 n5 ← n5 + 1;
12 ■
13 u ← (u*(nr mod 10)) mod 10;
14 ■
15 pentru i=1, n2-n5 executa
16 u ← (u*2) mod 10;
17 ■
18 scrie u;
19 stop.

```

2. Problema este simplă, se pot folosi vectori auxiliari fie cu elemente întregi, fie cu elemente de tip logic, definiți ca indici pe cifre:  $A[0..9]$ .

3. Se folosește formula (valabilă pentru  $p$  număr prim):

$$\left\lfloor \frac{n}{p} \right\rfloor + \left\lfloor \frac{n}{p^2} \right\rfloor + \left\lfloor \frac{n}{p^3} \right\rfloor + \left\lfloor \frac{n}{p^4} \right\rfloor + \dots + 0$$

Se descompune în factori primi numărul  $p$ , și se folosește formula pentru a determina cea mai mică putere a unui factor prim al său.

```

1  citeste n, p; max ← 0;
2  pentru i=2, √p executa
3  nr ← 0;
4  cat timp p mod i=0 executa
5  nr ← nr + 1;
6  p ← p div i;
7  ■
8  daca nr>0 atunci
9  j ← i; r ← 0;
10 cat timp j≤n executa
11 r ← r + n div j;
12 j ← j * i;
13 ■
14 daca max<r atunci max ← r;
15 ■
16 ■
17 ■
18 daca p>1 atunci
19 j ← p; r ← 0;
20 cat timp j≤n executa
21 r ← r + n div j;
22 j ← j * p;
23 ■

```

```

24      daca max<r atunci max ← r;
25      └─┐
26      └─┘
27  scrie r;
28  stop.

```

4. Se parcurge matricea și se pleacă de la presupunerea că orice element de valoare  $k$  reprezintă capătul unei piese valide. Se verifică toate cele 8 așezări ale piesei din punctul curent considerat (datorită simetriei).

5. Se sortează crescător după limita inferioară a intervalelor. La parcurgerea intervalelor se va actualiza cel mai mare capăt din dreapta ( $maxdr$ ). Orice interval pentru care capătul din dreapta este mai mic decât  $maxdr$  este redundant, deci inclus în altul.

6. Pentru a evita implementarea operațiilor pe numere mari (reținute cifră cu cifră în vectori), se memorează factorii primi pentru fiecare număr asociat unei parole. La final se verifică dacă au vreun factor prim comun.

7. Algoritmul va face mai multe parcurgeri prin elementele matricei, pentru fiecare moment identificându-se numărul de elemente care « se vor topi ». Pentru a nu număra de mai multe ori o poziție este indicat ca poziția care se transformă în aer la momentul curent să fie însemnată cu o valoare negativă, de exemplu.

8. Algoritmul are la bază principiul următor: pentru două șiruri cu  $n$  elemente, respectiv  $m$  ( $n < m$ ) sortate crescător, dacă primul are  $k$  elemente negative și  $p$  elemente pozitive ( $k+p=n$ ) se cuplează primele  $k$  elemente din primul șir cu primele  $k$  din al doilea și ultimele  $p$  elemente din primul șir cu ultimele  $p$  din al doilea șir.

```

1  integ n, m, a[100], b[100], i, j, r;
2  daca n>m atunci inverseaza a cu b;
3  sorteaza sirul a crescator;
4  sorteaza sirul b crescator;
5  r ← 0; i ← 0;
6  cat timp (a[i+1]<0)and(i+1<=n) executa i ← i + 1;
7  ──
8  pentru j←1, i executa r ← r + a[j]*b[j];
9  ──
10 pentru j←i+1, n executa r ← r + a[j]*b[m-n+j];
11 ──
12 scrie r;
13 stop.

```

9. Algoritmul presupune identificarea de  $k$  ori a elementului a cărui sumă a vecinilor este maximă la pasul respectiv. Pentru evitarea unor discuții suplimentare generate de numărul de vecini ai unui element, se poate inițializa linia și coloana 0, respectiv  $n+1$  cu valoarea 0.

**10. Problema cere o bună manipulare a structurilor de date, fiind avantajoasă preluarea datelor într-un tablou unidimensional cu elemente de tip înregistrare. Algoritmul presupune o etapă de ștergere a unui element dintr-un șir și căutarea subsecvenței de lungime maximă care poate fii realizată liniar.**

**11. Algoritmul are o complexitate pătratică și presupune parcurgerea în spirală a matricei.**

12. Algoritmul reprezintă o programare dinamică liniară. Se construiește o matrice cu două coloane în care elementele sunt în mulțimea {True, False} cu semnificația:

- $A(i,0)$  = True, dacă se poate obține un șir până în poziția  $i$  cu piesa  $i$  nerotită;
- $A(i,1)$  = True dacă se poate obține un șir până în poziția  $i$  cu piesa  $i$  rotită.

Relațiile de recurență se deduc imediat.

```

1  intreg n, i, j, p[100][1..2], t[100][0..1];
2  a[1][0] ← true; a[1][1] ← true;
3  pentru i=2, n executa
4      daca ((p[i][1]=p[i-1][2]) or (p[i][1]+p[i-1][2]=6)) and
5          (a[i-1][0]=true) atunci
6          a[i][0] ← true;
7          t[i][0] ← 0;
8      ■
9      daca ((p[i][1]=p[i-1][1]) or (p[i][1]+p[i-1][1]=6)) and
10         (a[i-1][1]=true) atunci
11         a[i][0] ← true;
12         t[i][0] ← 1;
13     ■
14     daca ((p[i][2]=p[i-1][2]) or (p[i][2]+p[i-1][2]=6)) and
15         (a[i-1][0]=true) atunci
16         a[i][1] ← true;
17         t[i][1] ← 0;
18     ■
19     daca ((p[i][2]=p[i-1][1]) or (p[i][2]+p[i-1][1]=6)) and
20         (a[i-1][1]=true) atunci
21         a[i][1] ← true;
22         t[i][1] ← 1;
23     ■
24 ■
25 i ← n;
26 daca a[n][0]=true atunci j ← 0
27     altfel j ← 1;
28 cat timp i>0 executa
29     daca j=1 atunci scrie p[i][1], p[i][2]
30     altfel scrie p[i][2], p[i][1];
31     j ← t[i][j];
32     i ← i - 1;
33 ■
34 stop.

```

13. Algoritmul este o programare dinamică în care se determină numărul de sume distincte ale submulțimilor care se pot forma cu elementele date. Se va crea un vector  $S$  de lungime  $N+1$  în care elementul  $S(i)=\text{True}$  înseamnă că suma de valoare  $i$  se poate obține cu elementele șirului inițial. Inițial  $S(0)=\text{True}$ .

14. Se determină şirul de numere super prime mai mici decât  $n$ , după care se determină cu ajutorul programării dinamice submulţimea de sumă  $n$  cu număr minim de elemente.

Se vor crea doi vector  $S(0..n)$  şi  $T(1..n)$ :

- $s(i)=j$  are semnificaţia suma  $i$  se poate obţine cu minim  $j$  termeni
- $t(i)=k$  are următoarea semnificaţie: suma de valoare  $i$  s-a obţinut cu ultimul termen  $a(k)$ .

```

1  întreg n, i, j, k, np, p[1000], nr[0..10000], t[0..10000];
2  logic ok;
3  k ← 0; np ← 0;
4  pentru i=2, n executa
5      ok ← true;
6      pentru j=2, √i executa
7          dacă i mod j=0 atunci ok ← false;
8      ■
9      dacă ok=true atunci
10         k ← k+1;
11         ok ← true;
12         pentru j=2, √k executa
13             dacă k mod j=0 atunci ok ← false;
14         ■
15         dacă ok=true atunci
16             np ← np + 1;
17             p[np] ← i;
18         ■
19     ■
20 ■
21 nr[0] ← 0;
22 pentru i=1, n executa
23     nr[i] ← ∞;
24 ■
25 pentru i=1, np executa
26     pentru j=0, n-p[i] executa
27         dacă nr[j+p[i]] > nr[j]+1 atunci
28             nr[j+p[i]] ← nr[j]+1;
29             t[j+p[i]] ← p[i];
30         ■
31     ■
32 ■
33 cat timp n>0 executa
34     scrie t[n];
35     n ← n - t[n];
36 ■
37 stop.

```

15. Algoritmul are la bază o programare dinamică în care este construită matricea  $B(n, m)$ . Elementul  $b(i, j)$  va reprezenta numărul maxim de mere care poate fi culese până în poziţia  $(i, j)$ . Elementul  $b(1, 1)$  va fi egal cu  $a(1, 1)$ ; Elementele primei linii se vor determina pe baza recurenţei  $B(1, j)=b(1, j-1)+a(1, j)$ . Elementele primei coloane se vor determina pe baza recurenţei  $B(i, 1)=b(i-1, 1)+a(i, 1)$ .

Relaţia de recurenţă generală:

$$b(i, j) = \max(b(i-1, j), b(i, j-1)) + a(i, j).$$

16. Problema se reduce la numărul de parantezări corecte pentru  $n$  perechi de paranteze, cunoscut şi ca numărul lui Catalan:  $(2n)! / (n! * n! * (n+1))$

17. Se vor identifica valorile distincte (fără repetiţii) din vectorul iniţial. Afişarea se poate face chiar şi fără folosirea unei matrici de caractere.

18. Se aplică algoritmul clasic pentru subşir crescător de lungime maximă folosind doar elementele care sunt numere prime.

```

1  întreg n, i, j, rez, poz, s[2000], nr[2000], t[2000];
2  logic ok;
3  rez ← 0; poz ← 0;
4  pentru i=1, n executa
5      ok ← true;
6      pentru j=2, √s[i] executa
7          dacă s[i] mod j=0 atunci ok ← false;
8      ■
9      dacă ok=true atunci
10         nr[i] ← 1;
11         t[i] ← 0;
12         pentru j=1, i-1 executa
13             dacă (nr[i] < nr[j]+1) and (s[j] < s[i]) atunci
14                 nr[i] ← nr[j]+1;
15                 t[i] ← j;
16         ■
17     ■
18 ■
19 altfel nr[i] ← 0;
20 ■
21 dacă rez < nr[i] atunci
22     rez ← nr[i]; poz ← i;
23 ■
24 ■
25 scrie nr[poz];
26 cat timp poz > 0 executa
27     scrie s[poz];
28     poz ← t[poz];
29 ■
30 stop.

```

//sunt afisate in ordine inversa

19. Vom nota cu  $D[i][j]$  = latura unui pătrat de arie maximă cu colţul dreapta-jos în  $(i, j)$ . Se deduce următoarea relaţie de recurenţă  $D[i][j] = 1 + \min(D[i-1][j], D[i][j-1], D[i-1][j-1])$  care se calculează doar dacă  $A[i][j] = 1$ .

```

1  întreg n, m, i, j, a[200][200], rez;
2  rez ← 0;
3  pentru i=1, n executa
4      pentru j=1, m executa

```

**20.** Vom nota cu  $D[i][j]$  = raza unui romb maxim cu colțul jos în  $(i, j)$ . Se deduce următoarea relație de recurență  $D[i][j] = 1 + \min(D[i-1][j+1], D[i-1][j-1], D[i-2][j])$  care se calculează doar dacă  $A[i][j] = A[i-1][j] = 1$ .

21. Pentru fiecare valoarea a primei secvențe (fie aceasta  $S_1$ ) se caută binar locul în care se împarte  $S_2$  și  $S_3$  în valori cât mai egale.

216

22. Se sortează şirul, şi apoi pentru fiecare valoare din şir a lui  $S[i]$  se caută binar valoarea  $M-S[i]$  de câte ori există.

23. Oricare patru puncte determină un punct de intersecție deci rezultatul va fi combinații de  $N$  luate câte 4.

**24.** Se construiește o matrice  $N \times K$  cu semnificația  $A[i, j]$  = numărul minim de zile necesare pentru a avea  $i$  persoane în camera și  $j$  perechi de persoane născute în aceeași zi. Există soluție doar dacă  $A[N, K] \leq Z$ . Relația de recurență va fi:

$$A[i, j] = \min (A[i-x, j-x^*(x-1)/2] + 1), x \leq i.$$

Pentru a reconstitui soluția se mai păstrează încă o matrice.

217

25. Se construiesc doi vectori cu următoarele semnificații:

- $A[i]$  = suma maximă a unui subșir cu elemente în primele  $i$  elemente, fără a folosi elementul  $i$ ;
- $B[i]$  = suma maximă a unui subșir cu elemente în primele  $i$  elemente, folosind elementul  $i$ .

Se deduc următoarele relații de recurență:  $A[i] = \max(A[i-1], B[i-1])$ ;  $B[i] = A[i-1] + S[i]$ . Răspunsul va fi  $\max(A[N], B[N])$ .

```
1:  intreg n, i, nr, a[0..1], b[0..1];
2:  citește n;
3:  pentru i=1, n executa
4:  |   citește nr;
5:  |   a[i] ← max(a[0], b[0]);
6:  |   b[i] ← a[0] + nr;
7:  |   a[0] ← a[i];
8:  |   b[0] ← b[i];
9:  |
10: |   scrie max(a[0], b[0]);
11: stop.
```

26. O proprietate necesară în rezolvarea problemei este următoarea: dacă cele  $M$  interschimbări sortează orice șir de lungime  $N$  cu valori 0 sau 1, sortează orice șir de numere. Fiind șiruri binare, se considera fiecare număr de la 0 la  $2^{N-1}$ , se decodifică și se aplică cele  $M$  interschimbări.

```
1:  intreg n, m, i, j, a[100], b[100], bit[18];
2:  citește m, n;
3:  pentru i=1, m executa citește a[i], b[i];
4:  |
5:  |   pentru i=0,  $2^n-1$  executa
6:  |   |   pentru j=0, n-1 executa bit[j+1] ← bitul j din numărul i;
7:  |   |   |
8:  |   |   |   pentru j=1, m executa
9:  |   |   |   |   dacă bit[a[j]] > bit[b[j]] atunci
10:  |   |   |   |   |   bit[a[j]] ← 0;
11:  |   |   |   |   |   bit[b[j]] ← 1;
12:  |   |   |   |
13:  |   |   |   |   pentru j=1, n-1 executa
14:  |   |   |   |   |   dacă bit[j] > bit[j+1] atunci
15:  |   |   |   |   |   |   scrie "NU";
16:  |   |   |   |   |   |   stop.
17:  |   |   |   |
18:  |   |   |   |   stop.
19:  |   |   |
20:  |   |   stop.
21: stop. scrie "DA"; stop.
```

27. Se construiește un vector  $\min[i]$  = numărul minim de timbre necesare pentru a obține  $i$  cenți. Relația de recurență este ușor de dedus.

28. Fie  $\phi(x)$  = numărul de numere mai mici ca  $x$  care sunt prime cu  $x$ . Răspunsul va fi  $1 + 2 * (\phi(2) + \phi(3) + \dots + \phi(N))$ . Pentru a calcula  $\phi(x)$  în mod eficient se poate folosi proprietatea  $\phi(p * q) = \phi(p) * \phi(q)$  dacă  $\text{cmmdc}(p, q) = 1$ .

29. Se observă că răspunsul se repetă din 100 în 100, deci este de ajuns calcularea restului lui  $N$  la 100 și preprocesarea răspunsurilor pentru valori de la 0 la 99.

30. Notăm cu  $A[i, j]$  numărul de permutări de lungime  $i$  cu  $j$  maxime. Se deduce următoarea relație de recurență:  $A[i, j] = A[i-1, j] * (i-1) + A[i-1, j-1]$ .

31. Este de ajuns să se verifice secvențele de lungime fix  $K$ . Se păstrează un vector  $Q$  care va conține indici din vector cu proprietatea că  $Q[i] \leq Q[i+1] \leq \dots$  și  $A[Q[i]] \leq A[Q[i+1]] \leq \dots$ . Se parcurge șirul iar când se ajunge la elementul de pe poziția  $x$  se elimină toate elementele de la sfârșitul lui  $Q$  mai mari ca valoare din vector decât  $A[x]$  și se inserează  $x$ . Dacă elementul de la început este pe o poziție mai mică decât  $x-K+1$  (nu este în secvența de lungime  $K$  ce se termină în  $x$ ) se elimină până când acesta va avea poziția mai mare sau egală cu  $x-K+1$ . Elementul de la început va fi baza secvenței de lungime  $K$  ce se termină în elementul  $x$ . Cum nu se fac mai mult de  $N$  eliminări și  $N$  inserări,  $Q$  va conține mereu cel mult  $N$  elemente distincte, complexitatea algoritmului fiind liniară în funcție de  $N$ .

```
1:  intreg n, k, i, st, dr, rez, poz, a[500000], q[500000];
2:  citește n, k;
3:  pentru i=1, n executa
4:  |   citește a[i];
5:  |
6:  |   rez ← ∞;
7:  |   st ← 1; dr ← 0;
8:  |   pentru i=1, k-1 executa
9:  |   |   cat timp (dr >= st) and (a[i] ≤ a[q[dr]]) executa dr ← dr-1;
10:  |   |   dr ← dr+1;
11:  |   |   q[dr] ← i;
12:  |   |
13:  |   |   pentru i=k, n executa
14:  |   |   |   cat timp (st ≤ dr) and (a[i] ≤ a[q[dr]]) executa dr ← dr-1;
15:  |   |   |   dr ← dr+1;
16:  |   |   |   q[dr] ← i;
17:  |   |   |   cat timp (st ≤ dr) and (q[st] < i-k+1) executa st ← st+1;
18:  |   |   |   dacă a[q[st]] > rez atunci
19:  |   |   |   |   rez ← a[q[st]];
20:  |   |   |   |   poz ← i;
21:  |   |   |
22:  |   |   stop.
23: stop. scrie poz-k+1, poz, rez;
```

32. Fie  $A$  matricea inițială, se construiește o matrice  $C[i, j]$  = diferența maximă de scor jucător 1 – jucător 2 (jucătorul 1 e cel ce începe din  $(i, j)$  cu jetonul).  $C[i, j] = A[i, j] - \max(C[x, y])$   $x \leq i$  și  $y \leq j$ . Pentru a calcula eficient  $C[i, j]$  se mai păstrează o matrice  $B[i, j] = \max(C[x, y])$   $1 \leq x \leq i$  și  $1 \leq y \leq j$ .

33. Se generează toate tripletele  $(i, j, k)$  cu  $i, j, k \leq N$  și se inserează într-un vector  $A[i] + A[j] + A[k]$ . Se sortează vectorul cu un algoritm eficient, iar pentru fiecare valoare din vector  $x$  se caută binar dacă există valoarea  $S-x$ .

34. Secvența de lungime minim  $K$  care se termină cu elementul de pe poziția  $i$  este fie secvența de pe poziția  $i-1$  + elementul  $i$  sau elementele  $i-K+1, i-K+2 \dots i$  (ultimele  $K$ ).

```
1  integ n, k, i, val, start, st, dr, rez, a[50000], s[50000];
2  citește n, k;
3  pentru i=1, n executa
4      citește a[i];
5      s[i] ← s[i-1] + a[i];
6  rez ← s[k];
7  st ← 1;
8  dr ← k;
9  val ← s[k];
10 start ← 1;
11 pentru i=k+1, n executa
12     val ← val+a[i];
13     dacă val<s[i]-s[i-k] atunci
14         val ← s[i]-s[i-k];
15         start ← i-k+1;
16     dacă rez<val atunci
17         rez ← val;
18         st ← start;
19         dr ← i;
20     scrie st, dr, rez;
21 stop.
```

35. Soluția se bazează pe proprietatea foarte importantă (subliniată și în enunț) că pe fiecare linie există un punct roșu și pe fiecare coloana un punct albastru. Presupunem că ținem o listă cu puncte.

Inițial introducem un punct roșu oarecare. Pe coloana punctului roșu respectiv există un punct albastru (din proprietatea de mai sus). Inserăm acel punct albastru în listă. Pe linia punctului albastru va exista un punct roșu, pe care îl vom insera în listă. Repetând acest procedeu vom ajunge la un moment dat la un punct care a mai fost în lista, deci la un ciclu (acest lucru este evident deoarece numărul punctelor este finit). Punctele roșii de pe ciclu vor reprezenta primul poligon, iar punctele albastre al doilea poligon.

Este evident că vor avea același număr de vârfuri, vom arăta în continuare că au și același centru de greutate.

Fie primul punct (acela roșu)  $(x_1, y_1)$ . Al doilea va fi  $(x_2, y_1)$ , al treilea  $(x_2, y_2)$ , al patrulea  $(x_3, y_2)$  .. penultimul  $(x_k, y_{k-1})$ , ultimul  $(x_k, y_k)$  (care va coincide cu un alt punct  $(x_p, y_p)$ ,  $p < k$ ).

Se observă că poligonul roșu va avea ca centru de greutate punctul  $((x_p + x_p + 1 + \dots + x_{k-1})/(k-p), (y_p + y_p + 1 + \dots + y_{k-1})/(k-p))$ , iar cel albastru  $((x_p + 1 + x_p + 2 + \dots + x_k)/(k-p), (y_p + 1 + y_p + 2 + \dots + y_k)/(k-p))$ , care coincid deoarece  $(x_p, y_p) = (x_k, y_k)$ .

36. Condiția ca oricare trei puncte nu sunt coliniare simplifică mult rezolvarea. Din definiție, un trapez are cel puțin două laturi paralele deci se poate construi următorul algoritm: se iau toate perechile de puncte - acestea determină câte un segment - și sortează în funcție de unghiul cu axa OX (panta dreptei). Pentru fiecare  $k$  segmente cu același unghi se pot forma  $k*(k-1)/2$  trapeze.

Pentru a evita calculele cu reale (care pot cauza erori de precizie), se țin pantele ca perechi de numere întregi  $(y, x)$ , fără a efectua efectiv împărțirea  $y/x$ . Pentru compararea a două astfel de perechi sunt necesare tipuri de date pe 64 de biți.

37. Rezolvarea se bazează pe programare dinamică. Vom numi cele două șiruri  $A$  și  $B$ , de lungime  $N$ , respectiv  $M$  și vom construi inițial matricea  $C[i][j]$  = lungimea celui mai lung subșir comun al șirurilor  $A[1..i]$  și  $B[1..j]$ . Acest lucru este o aplicație clasică a programării dinamice. Se va calcula o matrice  $Nr[i][j]$  = câte subșiruri comune de lungime maximă există pentru șirurile  $A[1..i]$  și  $B[1..j]$  (evident modulo 666013). Se calculează  $Nr[i][j]$  doar atunci când  $A[i] = B[j]$ , astfel: pentru fiecare caracter  $c$  între 'a' și 'z' se caută ultima sa apariție în șirul  $A[1..i-1]$  (fie aceasta poziția  $ii$ ) și ultima sa apariție în șirul  $B[1..j-1]$  (fie aceasta poziția  $jj$ ).

Dacă  $C[i][j] = C[ii][jj] + 1$  se va aduna  $Nr[ii][jj]$  la  $Nr[i][j]$  - aceasta condiție garantează că subșirurile adăugate au lungime maximă, iar faptul că  $ii$  și  $jj$  reprezintă ultima apariție a caracterului garantează că nu se vor număra subșiruri identice. Pentru a găsi rezultatul final se adună toate valorile  $Nr[i][j]$  calculate, cu următoarea excepție: dacă există pozițiile  $x$  și  $y$  astfel încât  $A[x] = A[i] = B[y] = B[j]$ , se adună  $Nr[i][j]$  doar dacă  $x < i$  și  $y < j$  (pentru a asigura ca nu se numără subșiruri identice de mai multe ori).

38. Răspunsul va fi al  $K+1$ -lea număr prim ridicat la pătrat. Se folosește ciurul lui Eratostene.

```
1  integ n, k, i, j; logic a[2000000];
2  citește k;
3  pentru i=2, 1000000 executa a[2*i] ← true;
4  i ← 3;
5  cat timp i≤2000000 executa
6      dacă a[i]=false atunci
7          k ← k-1;
8          dacă k=0 atunci
9              scrie i*i;
10             stop.
11     j ← i*i;
12     i ← i+1;
```

```

13 |   cat timp j ≤ 2000000 executa
14 |   a[j] ← true;
15 |   j ← j + 2*i;
16 |   ┌─┘
17 |   └─┘
18 |   i ← i + 2;
19 |   ┌─┘
20 |   stop.

```

39. Un număr  $xxx_{(b)}$  scris în baza 10 este  $nr = x \cdot b^2 + x \cdot b + x$ . Din rezolvarea inegalității de gradul 2,  $nr \leq N$  se deduce baza maximă în care pot fi scrise numerele căutate. Apoi se consideră toate numerele de forma cerută scrise în toate bazele mai mici sau egale ca limita determinată.

40. Se fixează mijlocul secvenței și se deplasează de la mijloc spre stânga și spre dreapta, comparând elementele simetrice. Se consideră ambele situații: lungime pară sau impară.

```

1 | 1. intreg n, i, st, dr, rez, start; sir s;
2 | 2. citeste n, s;
3 | rez ← 0;
4 | pentru i ← 1, n executa
5 |   st ← i;
6 |   dr ← i;
7 |   cat timp (st-1 > 0) and (dr+1 ≤ n) and (s[st-1] = s[dr+1]) executa
8 |   st ← st-1;
9 |   dr ← dr+1;
10 |   ┌─┘
11 |   └─┘
12 |   daca rez < dr-st+1 atunci
13 |     rez ← dr-st+1;
14 |     start ← st;
15 |   ┌─┘
16 |   └─┘
17 |   cat timp (st-1 > 0) and (dr+1 ≤ n) and (s[st-1] = s[dr+1]) executa
18 |   st ← st-1;
19 |   dr ← dr+1;
20 |   ┌─┘
21 |   └─┘
22 |   daca rez < dr-st+1 atunci
23 |     rez ← dr-st+1;
24 |     start ← st;
25 |   ┌─┘
26 |   └─┘
27 |   scrie start, rez;
28 | stop.

```

41. Se construiește vectorul sumelor parțiale  $S[i] = A[1] + A[2] + \dots + A[i]$  și se sortează. Apoi se iau elementele adiacente din vectorul sortat și se actualizează soluția.

```

1 | 1. intreg n, i, j, t, rez, a[100], s[0..100];
2 | 2. citeste n;
3 | pentru i ← 1, n executa
4 |   citeste a[i];
5 |   s[i] ← s[i-1] + a[i];
6 |   ┌─┘

```

```

7 | 7. sorteaza vectorul s crescator;
8 | rez ← s[1];
9 | pentru i ← 1, n-1 executa
10 |   daca rez > s[i+1]-s[i] atunci
11 |     rez ← s[i+1]-s[i];
12 |   ┌─┘
13 |   └─┘
14 | scrie rez;
15 | stop.

```

42. Se începe cu primul pitic: acesta este roșu dacă a răspuns 0 sau negru altfel. parcurg apoi piticii și se stabilește pentru fiecare dacă este roșu sau negru.

```

1 | 1. intreg n, i, r, poz, val, nr[20000];
2 | 2. citeste n;
3 | pentru i ← 1, n executa
4 |   citeste poz, val;
5 |   nr[poz] ← val;
6 |   ┌─┘
7 |   └─┘
8 |   r ← 0;
9 |   pentru i ← 1, n executa
10 |     daca nr[i] = r atunci
11 |       scrie 'R';
12 |       r ← r+1;
13 |     ┌─┘
14 |     └─┘
15 |     altfel
16 |       scrie 'N';
17 |     ┌─┘
18 |     └─┘
19 | stop.

```

43. Se sortează cele 3 șiruri descrescătoare și se încearcă toate cele 3<sup>3</sup> posibilități de aduna fiecare tip de bilă în primele 3 cutii ca mărime.

44. Se sortează punctele descrescător după x, iar la x egal descrescător după y. parcurg punctele și la fiecare pas se ține y-ul maxim întâlnit, astfel se determină dacă punctul curent este dominant sau nu.

```

1 | 1. intreg n, i, j, t, maxy, nr, x[10000], y[10000];
2 | 2. citeste n;
3 | pentru i ← 1, n executa
4 |   citeste x[i], y[i];
5 |   ┌─┘
6 |   └─┘
7 |   sorteaza punctele descrescator dupa x, iar la x egal
8 |   descrescator dupa y
9 |   nr ← 0; maxy ← -∞;
10 |   pentru i ← 1, n executa
11 |     daca y[i] > max atunci
12 |       nr ← nr+1;
13 |       maxy ← y[i];
14 |     ┌─┘
15 |     └─┘
16 |   scrie nr;

```



45. Se construiește un vector  $ok[i]$  = dacă se poate obține suma  $i$  cu elementele din vector; cu  $i$  între -10000 și 10000. Se caută apoi o valoare pentru care  $ok[i]$  este True, iar diferența absolută între  $i$  și suma elementelor- $i$  este minimă. Relațiile de recurență sunt ușor de dedus.

```

1  integ n, s, i, j, rez, a[100]; logic ok[-10000..10000];
2  citește n;
3  pentru i=1, n executa
4      citește a[i];
5      s ← s+a[i];
6
7  ok[a[1]] ← true;
8  pentru i=2, n executa
9      dacă a[i]<0 atunci
10         pentru j=-10000, 10000 executa
11             dacă (ok[j])and(j+a[i]≥0) atunci ok[j+a[i]]←true;
12
13         dacă a[i]>0 atunci
14             pentru j=10000, -10000 executa
15                 dacă (ok[j])and(j+a[i]≤10000) atunci ok[j+a[i]]←true;
16
17
18
19
20 rez ← ∞;
21 pentru i=0, s executa
22     dacă (ok[i])and(rez>|2*i-s|) atunci rez ← |2*i-s|;
23
24 scrie rez;
25 stop.

```

46. Când una din laturi e un număr impar se poate parcurge întreaga tablă în zigzag. Când ambele laturi sunt pare, se colorează tabla ca una de șah (pătratul (1, 1) este alb) și se alege minimul dintre pătratele negre. Acela va fi ocolit, în rest se va trece prin toate pătratele.

